НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 004.02

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
"___"_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп'ютерні науки»*

на тему: «*Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору*»

Виконав: студент *4* курсу, групи *ІС-52*
Поліщук Андрій Олегович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст.викл. Солдатова М.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст.викл. Халус О. А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Поліщуку Андрію Олеговичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору*»
керівник проекту *Солдатова М.О., ст. викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с
2. Термін подання студентом проекту “*03*”*червня* 2019 року
3. Вихідні дані до проекту
Технічне завдання
4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд аналогів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис масивів інформації
3. Математичне забезпечення: змістовна та математична постановки задачі, алгоритми пошуку оптимального шляху та опис методів розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення
5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту
5. Перелік графічного матеріалу
1. Схема структурна варіантів використання

2. Схема структурна компонентів програмного забезпечення

3. Схема структурна розгортання обчислювальних вузлів

4. Схема структурна класів програмного забезпечення

5. Схема структурна діяльності

6. Схема структурна послідовності

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«23» квітня 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*		
2.	*Аналіз існуючих методів розв'язання задачі*		
3.	*Постановка та формалізація задачі*		
4.	*Розробка інформаційного забезпечення*		
5.	*Алгоритмізація задачі*		
6.	*Обґрунтування використовуваних технічних засобів*		
7.	*Розробка програмного забезпечення*		
8.	*Налагодження програми*		
9.	*Виконання графічних документів*		
10.	*Оформлення пояснювальної записки*		
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ А.О. Поліщук

(підпис)

Керівник проекту _____ М.О. Солдатова

(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5221.1181-с.ПЗ	Пояснювальна записка	81	
2	А4	ДП ІС-5221.1181-с.ТЗ	Технічне завдання	10	
3	А4	ДП ІС-5221.1181-с.ПМВ	Програма та методика випробувань	14	
			Матеріали графічні		
4	А4	ДП ІС-5221.1181-с.ССВ	Схема структурна варіантів використань	1	
5	А4	ДП ІС-5221.1181-с.ССК	Схема структурна компонентів	1	
			програмного забезпечення		
6	А4	ДП ІС-5217.1181-с.ССР	Схема структурна розгортання обчислювальних вузлів	1	
7	А4	ДП ІС-5221.1181-с.ССК	Схема структурна класів програмного	1	
			забезпечення		
9	А3	ДП ІС-5221.1181-с.ССД	Схема структурна діяльності	1	
10	А3	ДП ІС-5221.1181-с.ССП	Схема структурна послідовності	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5221.1181-с.ВДП		
Розробн.		Поліщук А.О.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Солдатова М.О.					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, сімдесят вісім сторінок, сорок п’ять рисунків, чотирнадцять таблиць, один додаток, дев’яти джерел.

У дипломному проекті реалізована тема «Автоматизація пошуку оптимального шляху робота-рятівника в умовах невизначених площин». В дипломному проекті було розглянуто: найбільш унікальні та гнучкі методи пошуку оптимального шляху враховуючи отримані дані.

У розділі загальні положення був визначений процес діяльності для системи, варіанти використання, функціональні вимоги. Були визначені основні аналоги даної системи та наведений порівняльний аналіз. Була поставлена задача, визначені цілі та мета розробки.

У розділі з інформаційного забезпечення був визначений формат вхідних, вихідних даних та описані структури масивів інформації .

У розділі з математичного забезпечення було описано побудову структури для подальшого проектування руху роботу. Було розглянуто переваги та особливості алгоритмів пошуку оптимального шляху.

У розділі з програмного та технічного забезпечення визначені засоби розробки та технічні вимоги. Було обрано архітектуру програмного забезпечення відповідно до архітектури спроектовано діаграми класів, послідовності та компонентів.

У технологічному розділі наведена інструкція користувача та проведено тестування системи.

JAVA, JAVAFX, СЦЕНА, АКТОР, РОБОТЕХНІКА, ГРАФОВІ СТРУКТУРИ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of 6 sections, containing 40 figures, 6 tables, 1 application.

In the diploma project the theme "Automation of the search for the optimal path of robot-savior in conditions of uncertain planes" is realized. The diploma project considered: the most unique and flexible methods of finding the optimal path taking into account the data obtained.

In the General P rovisions section, a process has been defined for the system, usage options, functional requirements. The main analogs of this system were identified and a comparative analysis was presented. The task was set, the goals and purpose of the development were determined.

In the information support section, the input and output format was defined, an xml file was designed in accordance with the objectives of the project.

The section on mathematical support described the construction of the structure for the further design of the movement work. The advantages and features of algorithms for finding an optimal gap were considered, depending on the input data. The peculiarities of the algorithm for finding the total number of paths of the robot-saver to the given curvature were presented.

The software and hardware section defines the development tools and system specifications that the software will launch. The architecture of the software was chosen in accordance with the architecture of the class diagrams, sequences and components.

In the technology section user's manual was described and system was tested.

JAVA, JAVAFX, SCENE, ACTOR, ROBOTECHNICS, GRAPHIC STRUCTURES

ЗМІСТ

ВСТУП

Про актуальність даної бакалаврської роботи не доводиться говорити, оскільки безпека життю як рятувальних груп так і людей які опинились в катастрофі обернено пропорційна до часу, який використовують на пошук та прийняття рішення рятувальники.

Ми можемо спостерігати прориви в робототехніці для максимально спрощення життя людей, але завжди на першому місці було використання механічних роботів в місцях які недоступні людині через низку факторів.

11 вересня вперше були справді перевірені аварії рятувальних роботів. Після жахливого терористичного акту роботи були послані в завали, щоб шукати тих, хто вижив і тіла. Роботи мали проблеми з роботою в завалах Світового торгового центру і постійно застрявали або ламалися. З тих пір було сформовано багато нових ідей щодо рятувальних роботів[4]. Отже, через нестачу ряду випробовувальних актів роботи проявили себе не кращим чином, після чого через декілька років було створено R4 program, в яку входили вчені з різних країн світу для виявлення та усунення недоліків в рятувальних роботах.

Дана проблема виникає в багатьох науково-технічних завданнях, вирішення яких зводиться до задач на графових структурах. Вирішення даної задачі визначає найефективніший розподіл часу на подолання шляху, а це дає неабиякий прибуток на виробництві.

При розробці алгоритмів пошуку оптимального шляху основною проблемою, з якою зустрічається розробник, є визначення балансу між швидкодією та якістю програмного продукту.

Наступна проблема – продуктивність пристроїв. В реалізації розв’язку задачі ми зауважуємо, що кожний пристрій має коефіцієнт ефективності і враховуємо, що жодний пристрій не може бути більш ефективним за еталонний[4].

ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовища

Щорічно в світі відбувається понад десяток тисяч катастроф, порятунок людей прямо пропорційно залежить від оперативності рятувальних дій, однак далеко не завжди у рятувальників є можливість швидко діяти. Причиною цього є надзвичайна небезпека життю та людський фактор.

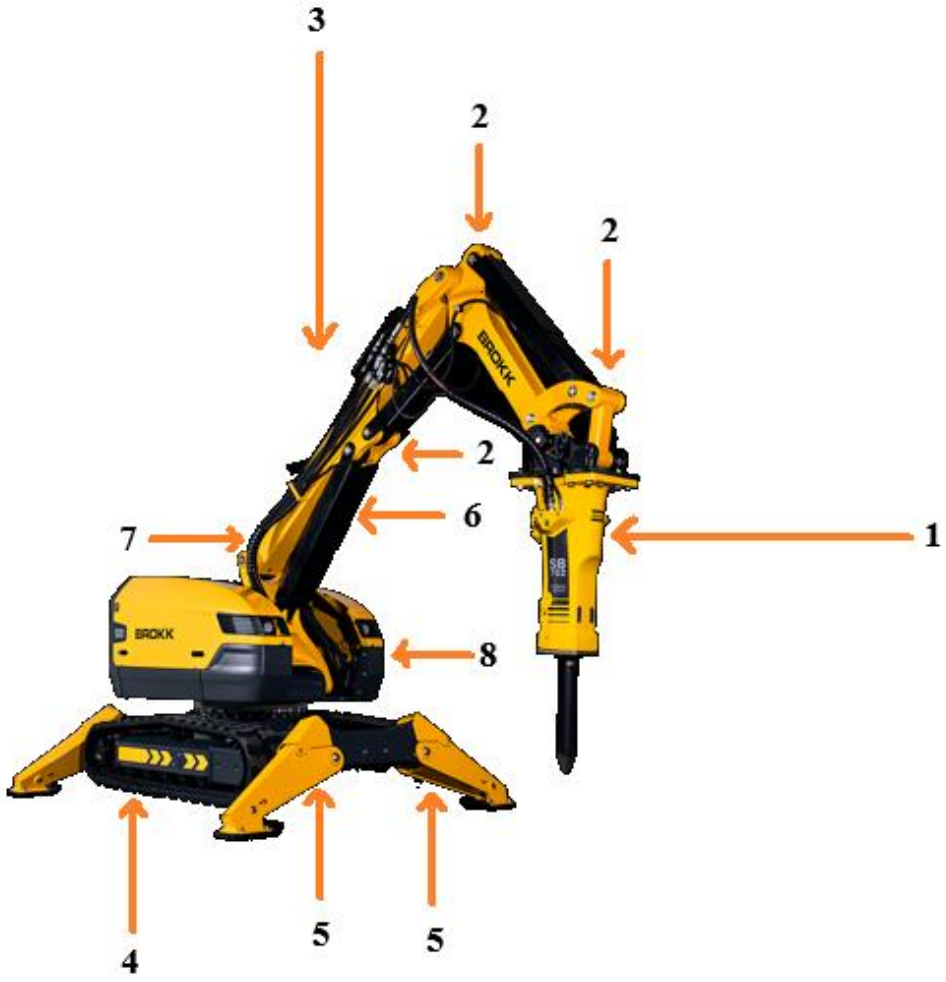


Рисунок 1.1 – Схематична модель робота рятувальника

На схематичному рисунку позначки означають: 1 – навісне обладнання, 2 – сегменти маніпулятора, 3 – маніпулятор, 4 – гусениці, 5 – виносні опори, 6 – гідроциліндр, 7 - вісь маніпулятора, 8 – датчик перешкоди[2].

Процес рятувальних дій зазвичай починається з аналізу плану та умов, які можуть впливати на безпеку та швидкодію рятувальних дій. Таким чином, постає необхідність розробки запрограмованого робота рятувальника, який би аналізуючи умови катастрофи за найкращий час і що є не менш важливим – безпечно для життя рятувальників провів рятувальні операції(пробурив) та звільнив би прохід під керівництвом рятувальних груп.

Для буріння товстих стін використовуються різні навісні обладнання, що збільшує швидкість буріння стін[2].

Рисунок 1.2 – Гідролом

Таким чином, постає необхідність розробки автоматизації робота, яка могла би пришвидшити та унеможливити смертельну небезпеку рятувальних дій. Таким чином, постає необхідність розробки автоматизації робота, яка могла би пришвидшити рятувальні дії. Програмний продукт повинен не тільки на основі вхідних даних знайти найкращий шлях та і підраховувати всі можливі альтернативі шляхи на випадок якщо пошукові дії продовжаться.

Рятувальні роботи використовувалися під час розшуку жертв і тих, що вижили після нападу 11 вересня в Нью-Йорку.

Рисунок 1.3 – Бетонолом

1.1.1. Опис процесу діяльності

Об’єктом автоматизації є процес пошуку роботом оптимального шляху для рятувальних робіт в умовах небезпечних для життя в замкнутому просторі типу будівлі. Розглянемо дії, які має виконати користувач для проведення рятувальної операції, за допомогою структурної схеми діаграми діяльності.

Схема структурна діяльності користувача наведена в графічному матеріалі.

1.1.2. Опис функціональної моделі

Виділимо акторів системи та опишемо дії, які може виконувати кожен актор у системі.

Із системою буде взаємодіяти один актор − користувач.

Користувач системи – це особа, яка може працювати з основними функціями системи.

Користувач. Клієнт, який використовуючи функціонал системи за допомогою інтерфейсу встановить всі налаштування робота та налаштує його роботу.

Схема структурна варіантів використання наведена в графічному матеріалі.

Виходячи з описаного функціоналу та можливостей, які має користувач, можна сформулювати діаграму варіантів використання.

1.1.3. Користувачі системи

Користувачем системи може бути людина, у якої є план структури, де проводяться рятувальні операції. Також варто розуміти, що користувачем системи може бути інший програмний продукт, який запрограмований на автоматичне налаштування робота рятівника.

1.2. Огляд наявних аналогів

В результаті пошуку аналогів були виявлені система зі схожими функціями: **SHEPRA, ICARUS, R4 PROGRAM**.

SHERPA

Метою SHERPA є розробка змішаної наземної та повітряної робото-технічної платформи для підтримки пошуково-рятувальних робіт у реальному середовищі, як у альпійському сценарії[1].

Технологічна платформа та сценарій альпійського порятунку є приводом для вирішення низки дослідницьких тем про пізнання та контроль, що стосуються виклику[1].

Те, що робить проект потенційно дуже багатим з наукової точки зору, є неоднорідністю та можливостями, якими володіють різні суб'єкти системи SHERPA: «людський» рятувальник - «зайнятий геній», який працює в

команді з наземним транспортним засобом, "розумний осел", і з повітряними платформами, тобто "навченими осами" і "патрулюючими яструбами". Дійсно, дослідницька діяльність зосереджена на тому, як «зайнятий геній» і «тварини SHERPA» взаємодіють і співпрацюють один з одним, зі своїми власними можливостями і можливостями, до досягнення спільної мети[1].

Поєднання передових засобів управління та когнітивних можливостей характеризує систему SHERPA, спрямовану на підтримку рятувальника шляхом підвищення його обізнаності щодо сцени порятунку навіть у важких умовах та з "генієм", який часто "зайнятий" у діяльності по рятуванню (і, таким чином, не може керувати платформи). Таким чином, акцент робиться на надійну автономність платформи, набуття когнітивних можливостей, стратегії співпраці, природну та неявну взаємодію між «генієм» і «тваринами SHERPA», які мотивують наукову діяльність[1].

ICARUS

Метою ICARUS є впровадження безпілотних пристроїв пошуку і порятунку може стати цінним інструментом для спасіння людських життів і прискорення процесу пошуку і порятунку. ICARUS концентрується на розробці безпілотних технологій SAR для виявлення, локалізації та порятунку людей[1].

Існує велика література з досліджень, спрямованих на розробку безпілотних інструментів пошуку і порятунку. Проте це дослідження стоїть на відміну від практичної дійсності на місцях, де безпілотні засоби пошуку та рятування рясніють шляхом пошуку кінцевих користувачів[1].

Проект ICARUS вирішує ці питання, спрямовані на подолання розриву між дослідницьким співтовариством та кінцевими користувачами, шляхом розробки інструментарію інтегрованих компонентів для безпілотного пошуку та порятунку[1].

R4 PROGRAM

Метою R4 program є тестування, впровадження та вдосконалення рятувальних роботів. П'ятнадцять вчених з усього світу були об'єднані в команду фахівців з пошуку та порятунку Федерального агентства з надзвичайних ситуацій Індіана. Вони були зібрані, щоб знайти проблеми з рятувальними роботами. Разом вони зібрали програму R4. Який є рятувальних роботів для досліджень і відповіді. Це трирічна дотація, і вона повинна поліпшити технологію рятувального робота і продуктивність людини. За цей час було протестовано трьох роботів, а вченим - четвертий. Кожен робот витрачав близько години, пересуваючись у завали, і спостерігався за їх рух і наскільки добре вони могли пробитися через руїни. Вони випробували роботів, що знаходяться на завали, від катастрофи Всесвітнього торгового центру, щоб вони могли краще підготуватися до подібної катастрофи. Вони шукали дві речі з цими рятувальними роботами. По-перше, як виявити жертв і небезпечні умови для рятувальників у сильно заплутаному, несприятливому середовищі. По-друге, як забезпечити охоплення датчика певного обсягу простору. В одній серії випробувань роботи були введені в темні, шахтні, подібні умови. Однак роботи не змогли виявити половину своїх цілей. Деякі зміни повинні бути зроблені, якщо вони коли-небудь очікують, що ці роботи будуть працювати належним чином. Але як тільки вони зрозуміють, що їм потрібно, вони сподіваються, що вони служать великій меті і будуть більшими активами для рятувальників[1].

Використання аналізу алгоритмів буде характерною відмінністю моєї розробки від розглянутих вище. Цілі розглянутих вище проектів пошуку та порятунку є ідентичними до використаних у моєму проекті, але функціонування, що використовують інші проекти зовсім різниться з використаним у цьому проекті.

В таблиці 1.1 наведено порівняльний аналіз основних важливих чинників вище перелічених систем.

Таблиця 1.1 - Порівняльний аналіз аналогів та даної роботи

	Дана робота	SHEPRA	ICARUS	R4 program
Наявність інтерфейсу для керування	Так	Так	Так	Ні
Оптимізація	Так	Ні	Ні	Так
Велика частина системи автоматизована	Так	Так	Так	Ні
Складна для розуміння користувачу	Ні	Так	Так	Так
Потрібні налаштування конфігурації	Так	Так	Так	Так
Об’єднує наземний та повітряний простір для пошуку	Ні	Так	Ні	Ні

1.3. Постановка задачі

1.4.1. Призначення розробки

Призначенням розробки є автоматизація робота рятівника для пошуку найбільш оптимального шляху порятунку за рахунок аналізу вхідних даних і вибору найкращого алгоритму в наслідок чого користувачу симулюється рух робота.

1.4.2. Мета та задачі розробки

Метою розробки є спрощення рятувальних операцій, забезпечення найбільш оперативних та безпечних дій знаходження шляху порятунку.

Для досягнення поставленої мети необхідно реалізувати наступні задачі:

- реалізувати алгоритми пошуку оптимального шляху робота рятівника;
- розробити систему вибору алгоритму пошуку оптимального шляху покладаючись на отриману інформації вхідних даних;
- розробити підсистему інтерфейсу, що змоделює для остаточного ухвалення план руху робота;
- розробити підсистему автоматичної генерації початкових умов для подальшого тестування застосунку клієнтами;
- розробити підсистему, яка буде працювати як і в двовимірному так і в тривимірному просторі(в перспективі розвитку робототехніки).

Висновок до розділу

В даному розділі було обґрунтовано доцільність створення системи автоматизації робота рятівника. Були висунуті основні вимоги до системи, сформульовано основні цілі та задачі розробки автоматизації пошуку оптимального шляху. Був проведений порівняльний аналіз між існуючими засобами та засобом, що реалізується в даній роботі в ході чого маємо очевидні перспективи та переваги над аналогами використання продукту в житті.

2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Вхідні дані

При автоматичній генерації стін та небезпечних ділянок для пошуку оптимального шляху роботом рятівником:

- довжина сторони будівлі;
- найменша товщина стіни;
- найбільша товщина стіни;
- координати кімнати початку пошуку(координата А);
- координати кімнати кінця пошуку(координата В);
- коефіцієнт частоти небезпечних ділянок переходу між кімнатами будівлі.

При отриманні даних з файлу для пошуку оптимального шляху роботом рятівником:

- довжина сторони будівлі, координати кімнати початку пошуку(координата А) та координати кімнати кінця пошуку(координата В) задаються в першому рядку файлу;
- перелік кожної наявної стіни зі вказаною товщиною стіни кожен наступний рядок файлу;
- перелік кожного наявного переходу між стінками зі вказанням чи небезпечний прохід кожен наступний рядок файлу після переліку списку товщин стін.

2.2. Вихідні дані

Візуальна демонстрація шляху та загально можливої кількості шляхів робота рятівника в інтерфейсі користувача.

Оптимальний шлях робота рятівника та його повна довжина в вихідному файлі.

2.3. Структура масивів інформації

Відображаємо структуру даних (масивів інформації) у вигляді XML-файлу:

<ArrayOfInformation>
<title> AlgorithmCell</title>
<package> algorithm </ package >
<class> HexagonAlgorithmField </class >
<semantics> Зберігається кінцевий варіант алгоритму для 3D моделі</semantics>
</ArrayOfInformation>

<ArrayOfInformation>
<title> AlgorithmCubicle</title>
<package> algorithm </ package >
<class> SphereAlgorithmField</class >
<semantics>Зберігається кінцевий варіант алгоритму для 2D моделі</semantics>
</ArrayOfInformation>

<ArrayOfInformation>
<title> ofVisit</title>
<package> algorithmMinPath</ package >
<class> AntAlgoritm</class >
<semantics>Зберігаються координати кімнати, які було відвідано</semantics>
</ArrayOfInformation>

<ArrayOfInformation>

<title> arrayOfPheromones</title>

<package> algorithmMinPath</ package >

<class> AntAlgoritm</class >

<semantics>Зберігаються позначки на кімтанах, які в подальшому вказуватимуть на скільки ефективний перехід між кімнатами</semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> finishArrayOfPathWays</title>

<package> algorithmMinPath</ package >

<class> AntAlgoritm</class >

<semantics>Зберігається остаточний варіант координат руху робота</semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> etaForTheCurrentPoint</title>

<package> algorithmMinPath</ package >

<class> AntAlgoritm</class >

<semantics>Зберігається актуальний на задану одиницю часу η(эта) - коефіцієнт </semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> pathLength</title>

<package> algorithmMinPath</ package >

<class> AntAlgoritm</class >

<semantics>Зберігається довжина шляху робота </semantics>

</ArrayOfInformation>

<ArrayOfInformation>
<title> matrixOfShortestArcs</title>
<package> algorithmMinPath</ package >
<class> Dijkstra</class >
<semantics>Зберігається матриця досяжності </semantics>
</ArrayOfInformation>

<ArrayOfInformation>
<title> auxiliaryArray</title>
<package> algorithmMinPath</ package >
<class> Dijkstra</class >
<semantics> Зберігається остаточний варіант координат руху робота </semantics>
</ArrayOfInformation>

<ArrayOfInformation>
<title> walls</title>
<package> model</ package >
<class> FieldGenerator2D</class >
<semantics> Зберігаються автоматично створені товщини стінок </semantics>
</ArrayOfInformation>
<ArrayOfInformation>
<title> walls</title>
<package> model</ package >
<class> Field2D</class >
<semantics> Зберігаються товщини стінок зчитані з файлу </semantics>

</ArrayOfInformation>

<ArrayOfInformation>
<title> rooms</title>
<package> rooms </ package >
<class> Field2D</class >
<semantics> Зберігаються товщини стінок зчитані з файлу </semantics>
</ArrayOfInformation>

ArrayOfInformation>
<title> hazardousAreas</title>
<package> model</ package >
<class> Field2D</class >
<semantics> Зберігаються небезпечні ділянки для переходу зчитані з файлу </semantics>
</ArrayOfInformation>

<ArrayOfInformation>
<title> rooms</title>
<package> model</ package >
<class> FieldGenerator2D</class >
<semantics> Зберігаються автоматично створені координати кімнат </semantics>
</ArrayOfInformation>

ArrayOfInformation>
<title> hazardousAreas</title>
<package> model</ package >
<class> FieldGenerator2D</class >

<semantics> Зберігаються автоматично створені небезпечні ділянки для переходу </semantics>

</ArrayOfInformation>

ArrayOfInformation>

<title> walls</title>

<package> model</ package >

<class> Field3D</class >

<semantics>Зберігаються товщини стінок зчитані з файлу </semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> fieldSceneHolder</title>

<package> util</ package >

<class> RobotAplication</class >

<semantics> Зберігаються інформація про зміни користувача на “сцені”</semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> IntroductionSceneHolder</title>

<package> visualization</ package >

<class> IntroductionSceneHolder </class >

<semantics> Зберігаються початкові налаштування інтерфейсу користувача</semantics>

</ArrayOfInformation>

<ArrayOfInformation>

<title> newRibs</title>

<package> algorithmMinPath</ package >

<class> PreparationForDijkstra3D</class >

<semanticsЗберігаються трансформовані переходи між кімнатами для пощуку оптимального шляху</semantics>

</ArrayOfInformation>

Висновок до розділу

В даному розділі було обумовлено вхідні та вихідні дані та їх типаж. Було описано структури інформації в вигляді XML розмітки, де вказувалось найменування масиву інформації, пакет, де знаходиться дана структура, java-клас, та його семантика, для розуміння того для чого використовуються дані. Була зображена xml–схема документу для розуміння взаємозв'язків між елементами xml-файлу.

3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Змістовна постановка задачі

Процедура вибору найкращого шляху роботом рятівником до таких ключових аспектів:

- створення та динамічний аналіз алгоритмів пошуку оптимального шляху;

- створення локальної зони пошуку в умовах невизначеної площини шляхом формування структури, де проходять рятувальні операції, аналогічно плану будівлі.

Розглянути можливість руху робота рятівника як в двомірному так і в тривимірному просторі для збільшення швидкодії програми.

В даній системі можемо обумовити ряд основних акторів на сцені:

- контролер генерації та демонстрації користувачу структури по якій буде рухатись робот рятівник на основі вхідних даних;

- навігатор новоствореної структури;

- аналізатор алгоритмів;

- перемикач, що вказує вимірність для генерування вище обумовленої структури;

Виходячи з цього маємо ряд важливих математичних задач, вирішення яких гарантує якісний, зрозумілий користувачу та максимально швидкий результат.

Задача створення генератора структури вартує доволі великих зусиль, оскільки для виконання поставленої задачі потрібно правильно провести моделювання графа по якому буде рухатись робот рятівник. Проектувати структуру слід враховуючи вимірність простору, а також для найшвидшого

проведення рятувальної операції після отримання вхідних даних слід сформувати фективні «стіни» з нульовою товщиною для побудови відповідної структури.

Не менш складною є задача створення найбільш гнучких та швидких алгоритмів пошуку та динамічного аналізатора, який за бажанням користувача проведе аналіз вхідних даних на основі чого буде автоматично обрано алгоритм пошуку.

Досить тривіальною задачею є створення алгоритму пошуку кількості всіх можливих шляхів пошуку відповідно до заданої «кімнати» пошуку.

3.2. Математична постановка задачі

3.2.1. Задача створення формалізованої структури для можливості проектування пошуку оптимального шляху робота рятівника

Основним призначенням цієї задачі є створення формалізованого графу по якому буде рухатись робот рятівник. Від час відповідного етапу роботи алгоритмів генерації структура представлена відповідною колекцією. Дано:

- довжина сторони структури, в якій будуть згенеровані «кімнати» та перехід між ними. У випадку якщо під час формування структури будуть відсутні переходи, то формуватимуться фективні переходи
- координати вершини графу звідки починає пошук та координати вершини графу де закінчує пошук робот рятівник
- товщину «стін» для переходу між «кімнатами» - вартість переходу між вершинами графу.
- інформацію про завалені переходи між «кімнатами».

Ціль задачі – оперуючи даними параметрами згенерувати колекцію, елементи якої відповідають формалізованому графу, що представляє собою набір вершин – «кімнат» та переходів між ними «стінок».

3.2.2. Задача пошуку оптимального шляху методом використання алгоритму Дейкстри

Призначенням цієї задачі є пошук найкращого шляху шляхом алгоритму Дейкстри. Відповідно до сформованої структури нам відомо:

- координати вершини графу звідки починає пошук та координати вершини графу де закінчує пошук робот рятівник
- координати вершин графу та вартості переходу між ними, що містить в собі створена структура.

Ціль задачі – використовуючи вхідні дані отримати найменш короткий шлях за найменший час виконання алгоритму.

3.2.3. Задача пошуку оптимального шляху методом використання мурашиного алгоритму

Призначенням цієї задачі є пошук найкращого шляху використовуючи бджолиний алгоритм. Відповідно до сформованої структури нам відомо:

- координати вершини графу звідки починає пошук та координати вершини графу де закінчує пошук робот рятівник
- координати вершин графу та вартості переходу між ними, що містить в собі створена структура.

Ціль задачі – використовуючи вхідні дані отримати найменш короткий шлях опираючись на одразу на фактор завалу проходів та на фактор товщин стін.

3.2.4. Задача пошуку загально можливої кількості шляхів

Призначенням цієї задачі є пошук загальної кількості шляхів робота рятівника до цілі. Відповідно до сформованої структури нам відомо:

- координати вершини графу звідки починає пошук та координати вершини графу де закінчує пошук робот рятівник.

- координати вершин графу та вартості переходу між ними, що містить в собі створена структура.

3.3. Обґрунтування методу розв’язання

Задача 3.2.1 може бути вирішена шляхом генерації колекції та запису в неї згенерованих даних відповідно до вимірності структури та спроектованої структури для точної та швидкої роботи робота рятівника.

При вирішенні задачі 3.2.2 потрібно вдосконалити алгоритм Дейктри шляхом щоб в результаті роботи ми отримували не тільки довжину шляху, що є класичним варіантом даного алгоритму, а й самі координати шляху між вершинами графу для руху робота рятівника.

При вирішені задачі 3.2.3. використовуючи бджолиний алгоритм, що є еврістичним, потрібно врахувати як і вагу переходу між вершинами графу так і завали, що унеможливлюють певні проходи.

Вирішуючи задачу 3.2.4. ми можемо використати рекурсивну функцію, що підраховуватиме кількість можливих шляхів для проходу робота рятівника шляхом суми шляхів, що межують з заданою вершиною.

3.4. Опис методів розв’язання

3.4.1. Побудова формалізованої структури на основі вхідних даних

В основу покладено рекурсивна генерація структури графу.

Для 2D

Структура має форму правильного шестикутника. Робот може рухатися в будь-яку з сусідніх кімнат(рисунок 3.1). Знайти шлях від кімнати A до кімнати B, при якому сумарна товщина пробурених стін буде найменшою.

Рисунок 3.1 - Система координат для правильного шестикутника

Для 3D

Структура складається з вершин графу, що мають форму шару(рисунок 3.2).

Рисунок 3.2 - Система координат для руху робота рятівника в 3D моделі

Доступні координати для переходу із координати (i,j,k) вершин графу(рисунок 3.3)

Рисунок 3.3 - Всі можливі координати для переходу із вершини графу i,j,k

Крок 1. Згенерувати колекцію для запису кожної кординати структури та вартості переходу між цими координатами.

Крок 2. Записати в колекцію координати вершин та вартість переходу.

Крок 3. Виконувати крок 2 до того часу, поки не перебрані всі ребра заданої структури:

3.4.2. Пошук оптимального шляху використовуючи алгоритм Дейкстри

Спочатку отримаємо на основі сформованої колекції шляхом парсингу дані про вершини графу та ваги дуг між ними[7]. Сформуємо матрицю суміжності G.

Маркований граф	Матриця суміжності
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ Координати 1-6.

Рисунок 3.3 – Демонстрація побудови матриці суміжності

Крок 1.

Проініціалізуємо всі відстані від початкової заданої вершини як $v = \infty$, де v – відстань від початкової вершини до всіх інших, що суміжні. Відстань до самої себе вершина ініціалізує як 0, марка поточної вершини ініціалізується як 0(рисунок 3.4)[7].

Рисунок 3.4 – Початковий стан графу

Крок 2.

Якщо при аналізі відстань від поточної вершини до суміжних менший ніж поточний мінімальний шлях, то змінюємо марку цієї вершини на відповідну довжину до цієї сусідньої вершини(рисунок 3.5)[7].

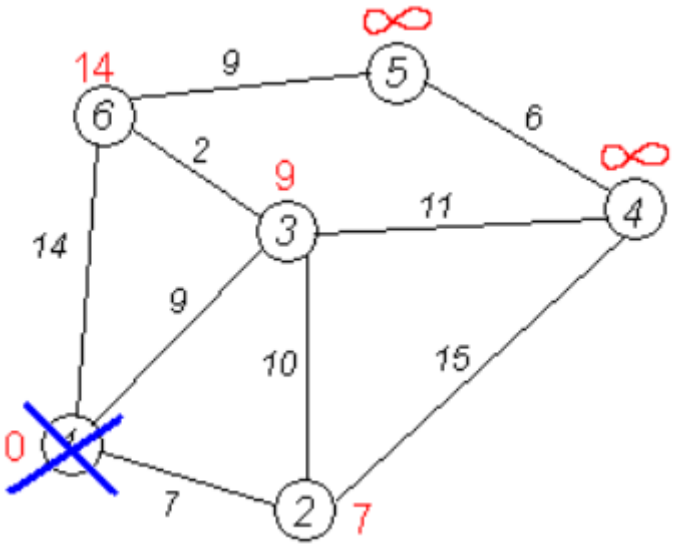


Рисунок 3.5 – Зображення міток сусідніх вершин до початкової вершини після

Крок 3.

Оберемо наступну вершину шляхом пошуку найбільш вигідного переходу, якщо марка + довжина ребра мінімальні з усіх можливих переходів поточної вершини і ця вершина не відвідувалась, то переходимо в знайдену вершину(рисунок 3.6) [7].

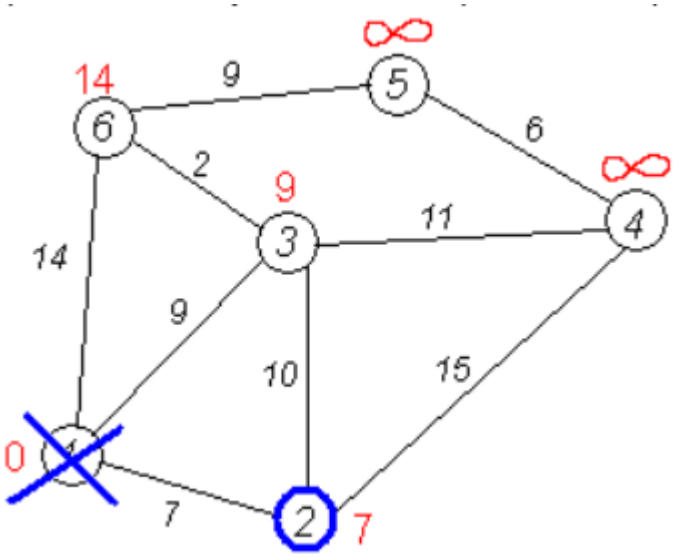


Рисунок 3.6 – Стан графу після закінчення обробки міток від початкової вершини

Крок 4.

Повторюватимемо кроки 2-3 поки всі вершини графу не будуть відвідані. Після чого отримаємо найкоротший шлях до всіх вершин графу від заданої вершини(рисунок 3.7) [7].

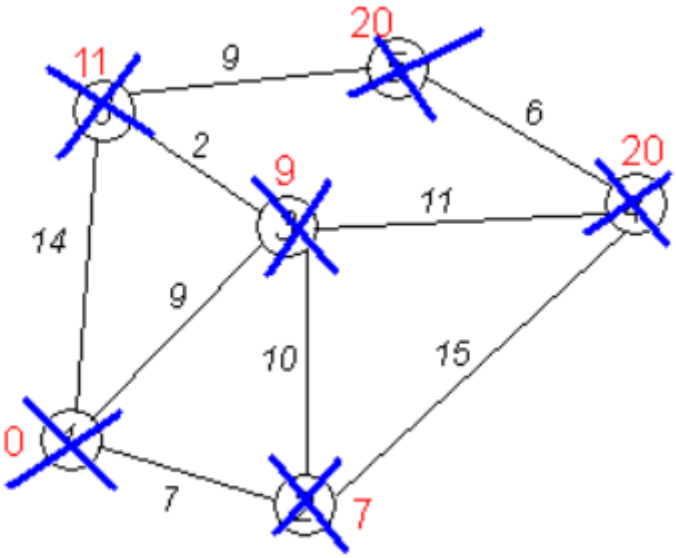


Рисунок 3.7 – Кінцевий стан графу після відвідання усіх вершин алгоритмом Дейкстри

3.4.3. Пошук оптимального шляху використовуючи мурашиний алгоритм

Мураха знаходиться в початковій вершині графу заданій у вхідних даних. Для визначення вершину в яку мураха перейде використаємо формулу[6]:

$$P_i = \frac{{l_i}^q \cdot {f_i}^p}{\sum_{k=0}^{N} {l_k}^q \cdot {f_k}^p}$$, де

P_i - ймовірність переходу шляхом i,

l_i - довжина i-ого переходу,

f_i - кількість феромонів на i-ому переході,

q **-** величина, яка визначає «жадібність» алгоритму**,**

p **-** величина, яка визначає «стадність» алгоритму.

Після кожного повного пройденого циклу по всім вершинам графу відбувається підрахунок залишеного феромону, який був залишений на кожному ребрі мурашкою[6].

$$\triangle \tau_{ij}^k(t) = \frac{Q}{L^k(t)}$$, де

Q – задана константа.

В результаті чого ми матимемо вимір шляху, чим коротший шлях, тим більше значення феромону. Після чого ми збільшимо феромони вздовж всього пройденого мурахою шляху використовуючи формулу[6]:

$$\tau_{ij}(t) = \triangle \tau_{ij}(t) + (\tau_{ij}^k \times \rho)$$

Значення змінної p – є константним і варіюється між значенням 1 та 0.

Для поступового очищення затратних в плані ресурсів ребер, що відносяться до найгірших шляхів спостерігається випаровування феромонів за допомогою рівняння:

$$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho)$$

3.4.4.Пошук загально можливої кількості шляхів робота рятівника до заданої координати

Крок 1.

Проініціалізуємо матрицю відвідувань символами ∞. Задамо початковій координаті значення загальної кількості шляхів 1. Перейдемо до наступної координати. Відмітимо в матриці відвідувать поточну вершину.

Крок 2.

Якщо вершина не відвідувалась, то перейдемо в обумовлену вершину. Підрахуємо кількість можливих шляхів робота рятівника шляхом сумування кількості можливих шляхів(рисунок 3.3 та рисунок 3.8) вершин з яких можна потрапити в поточну вершину. Відмітити поточну вершину в матриці відвідувань.

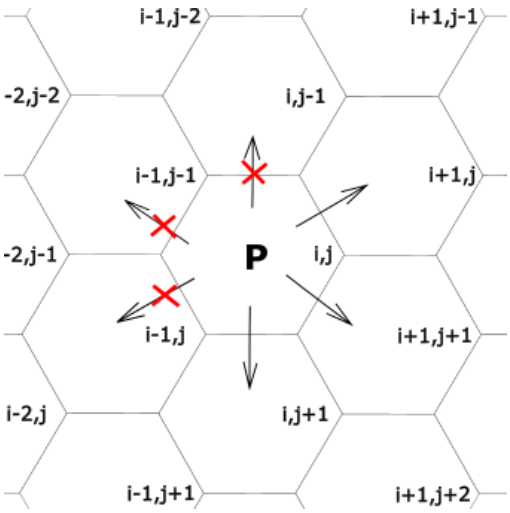


Рисунок 3.8 – Можливі напрямки руху робота для 2D

Крок 3.

Повторювати крок 2 допоки задана користувачем кінцева вершина не буде відвідана . Відмітити поточну вершину в матриці відвідувань.

Висновок до розділу

В даному розділі сформульовано математична постановка задачі та докладно розглянуто моделювання структури заданої користувачем. В якості алгоритмів пошуку оптимального шляху роботом рятівником було запропоновано точний алгоритм Дейкстри та еврістичний мурашиний алгоритм. А також було наведено детальний опис алгоритмів розв’язання поставлених задач.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Засоби розробки

Для розробки програмного продукту було обрано фреймворк розробки настільних застосунків JavaFX з використанням найпоширенішим архітектурним шаблоном для даної платформи MVVM[5]. Його основними перевагами є:

- широка підтримка платформи
- продуктивність
- велика стороння підтримка (інструменти, елементи управління і т.д.)

Так, шаблон архітектури Model-View-ViewModel (MVVM) розділяє продукт на три основні компоненти: модель, представлення та логіку представлення[5].

MVVM являє собою стандартний шаблон розробки, який походить від шаблонів MVP та MVC.

Однією з найголовніших переваг JavaFX є створена розробниками та задекларована мова «візуального дерева»[9], що не є дочірнім елементом xml. Інтерфейс користувача підв’язаний на графіці сцени, що дає можливість включати в розробку анімацію та різні ефекти для покращення розуміння результату виконання програми[8].

Для написання коду було обрано мову Java. Застосунок може працювати на будь-якому, де присутня JVM.

Java – це єдина мова програмування, яка реалізує платформу та набір інструментів JavaFX. Java є стандартом для корпоративних обчислювальних систем. Одною із найвагоміших переваг Java є незалежність від платформи, створивши Java-застосунок на Windows після компіляції в байт-код можемо запускати застосунок на будь-якій платформі, що підтримує JVM[9].

JVM – це так звана віртуальна машина Java, яка є основною виконуючою частиною системи Java. В результаті чого за рахунок віртуальної машини компілюється байт-код, що створений в редакторі(рисунок 4.1)[8].

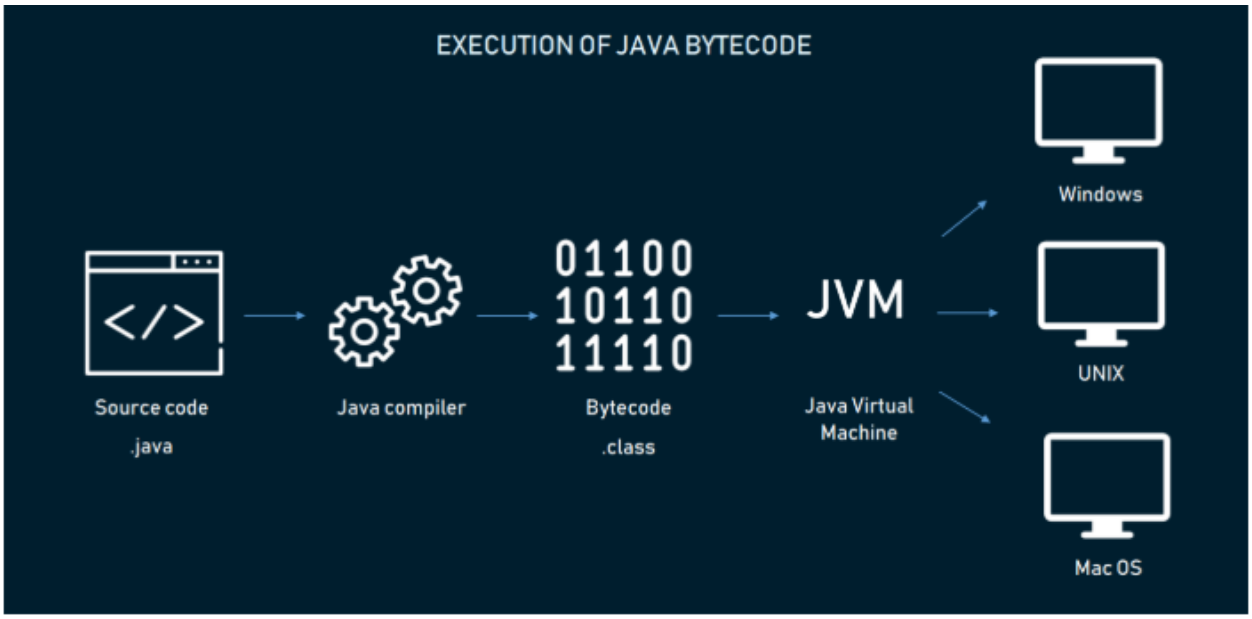


Рисунок 4.1 – Життєвий цикл програми

4.2. Вимоги до технічного забезпечення

Оскільки спроектована система є реалізацією використання платформи та інструментарію JavaFX, то вимоги пов’язанні з технічним забезпеченням повинні відповідати вимогам до цієї платформи та мови програмування Java.

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити процедуру генерації зруйнованої будівлі вказаного розміру та складності.

Для правильної роботи розробленої системи до складу технічних засобів потрібен комп’ютор з наступними вимогами:

- конфігурація комп’ютора:

1) двох ядерний процесор з частотою не нижче 2.5 ГГц;

2) достатній об’єм оперативної пам’яті (не менше 4 ГБ);

3) відеокарта з об’ємом пам’яті не менше 2048 Мб та частотою графічного процесора 1006 МГц.

- програмне забезпечення:

Необхідним обладнанням для роботи системи є комп’ютер зі встановленим середовищем Java Runtime Environment(jre), яка необхідна для виконання Java-застосунків, без компілятора та інших середовищ розробки та операційна система Windows 7, Windows 8 та Windows 10 64-bit або Mac OS X 10.9.2.

JRE містить в собі віртуальну машину – Java Virtual Machine і бібліотеки Java-класів.

- комп’ютерна периферія, до складу якої входить:

1) монітор;

2) комп’ютерна миша;

3) клавіатура.

4.3. Архітектура програмного забезпечення

4.3.1. Діаграма класів

Схема структурна класів, які реалізують вимоги відповідно до поставленої задачі, наведена в графічному матеріалі.

Діаграма класів може включати в себе статичні елементи типу не тільки класів та відношення між ними, а й інтерфейси та вкладенні пакети.

В поточному випадку зображено шістнадцять класів та один інтерфейс:

– «RobotApplication» - клас програми, що відповідає за відкриття початкової сторінки інтерфейсу та переходу на наступну сторінку;

– «IntroductionSceneHolder» - клас програми, що відповідає за оформлення початкової сторінки та розміщення елементів на ній;

– «FieldSceneHolder» - клас програми, що відповідає оформлення головної сторінки, де проводиться налаштування конфігурації робота, відподіває за розміщення елементів на сторінці, відповідає за обробку подій на «сцені»;

– «SceneHolder» - інтерфейс програми, що відповідає за створення «сцени»;

– «VisualField2D» - клас програми, що відповідає за генерація та відображення користувачу формалізованої структури для руху робота в двовимірному просторі;

– «VisualField3D» - клас програми, що відповідає за генерація та відображення користувачу формалізованої структури для руху робота в тривимірному просторі;

– «VisualCell» - клас програми, що відповідає за генерація та відображення користувачу оптимізованої навігації по створеній структурі в двовимірному просторі;

– «VisualCubicle» - клас програми, що відповідає за генерація та відображення користувачу оптимізованої навігації по створеній структурі в тривимірному просторі;

– «AlgorithmCell» - клас програми, що відповідає за підрахунок всіх можливих шляхів до заданої координати структури;

– «Cell» - клас програми, що передає інформацію про початкову координату робота рятівника та його координату пошуку двовимірній структурі, а також повертає значення кількостей шляхів до поточної вершини;

– «HexagonAlgorithmField» - клас програми, що оброблює інформацію про створену структуру, та записує в колекцію кількість можливих переміщень до кожної координати структури в двовимірному просторі;

– «AlgorithmCubicle» клас програми, що передає інформацію про початкову координату робота рятівника та його координату пошуку тривимірній структурі, а також повертає значення кількостей шляхів до поточної вершини;

– «SphereAlgorithmField» клас програми, що оброблює інформацію про створену структуру, та записує в колекцію кількість можливих переміщень до кожної координати структури в тривимірному просторі;

– «AntAlgorithm» клас програми, що реалізує мурашиний алгоритм;

– «Dijkstra» клас програми, що реалізує алгоритм Дейкстри;

– «PreparationForDijkstra2D» клас програми, що створення структур масивів інформації для подальшого використання в алгоритмі Дейкстри для двовимірного простору;

– «PreparationForDijkstra3D» клас програми, що створення структур масивів інформації для подальшого використання в алгоритмі Дейкстри для тривимірного простору;

4.3.2. Діаграма послідовності

Автоматизований пошук оптимального шляху робота рятівника починається з генерації структури обраної користувачем вимірності тому об’єкти які відповідають за підготовку до виконання пошуку та створюють відповідні структури отримають повідомлення про створення. Після завершення підготовчих дій ми отримуємо відповідну структуру наповнену формалізованими даними.

Після серії повідомлень за допомогою визначених алгоритмів обраховуємо довжину шляху, координати самого шляху та загально можлива кількість шляхів до заданої користувачем координати.

Рисунки схеми структурної послідовності взаємодії об’єктів наведено в графічному матеріалі.

4.3.3. Діаграма компонентів

Для ефективної реалізації продукту було побудовано діаграму з використанням моделі MVVM.

Схема структурна компонентів системи наведена в графічному матеріалі.

В зображеній системі можна виділити три найголовніших компонента таких як View, Model та ViewModel.

View являє собою графічний інтерфейс, що включає в себе різноманітні графічні об’єкти. View включає в себе компоненти: VisualField2D, VisualField3D, VisualCell та VisualCubicle. Дані компоненти відповідають за оформлення, розміщення елементів та перехід між «сценами».

Model включає в себе всі елементи, що являють собою побудову моделі. Компоненти, що входять в поточну групу вираховують оптимальних шлях використовуючи різні алгоритми, будують структури для руху робота рятівника, підраховують загальну можливу кількість шляхів до заданої координати. Тобто Model це набір фундаментальний даних, що необхідні для розробки застосунку

ViewModel – так звана модель, що перетворена до вигляду. Зазвичай містить в собі команди, що можуть впливати на модель. В поточному випадку включає в себе такі компоненти: Field2DSceneHolder, IntoductionSceneHolder, SceneHolder, ImageFillingRegion. Дані елементи відповідають за взаємозв’язок між інтерфейсом та обчислювальними процесами.

Висновок до розділу

В даному розділі було визначено засоби розробки, розглянуто особливості та переваги методів розробки та мови програмування на якій працює платформа та інструментарій JavaFX. Було вказано вимоги до технічного забезпечення та до обладнання відповідно до вимог інструментарію JavaFX.

Було наведено та описано діаграму класів, а також наведено діаграму послідовності з описом елементів, яка відображає взаємодію об’єктів впорядкованих часом, а також діаграму компонентів в вигляді MVVM для наглядності.

5. ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1. Керівництво користувача

В результаті створення програмного продукту за допомогою платформи JavaFX та її інструментарію створено такий функціонал:

- Генерація формування типізованої структури для моделювання руху робота рятівника;
- Генерація пошуку оптимального шляху алгоритмом Дейкстри;
- Генерація пошуку оптимального шляху мурашиним алгоритмом;
- Генерація пошуку загальної кількості шляхів до заданих координат, які може пройти робот рятівник;
- Генерація «сцени» та її елементів управління.

Опишемо акторів сцени з якими веде взаємодію користувач системи:

- Field Scene Hodler – надає взаємозв’язок між інтерефейсом та моделю. Слугує для управління обробників подій на головній «сцені» та оформленням цієї «сцени». Відповідає за генерацію «сцени» та її елементів.
- Algorithm Cell – відповідає за генерацію пошуку кільеості шляхів до заданих користувачем координат, які може пройти робот рятівник;
- VisualCubicle та VisualCell – відповідає за навігацію користувача по створеній структурі;
- Introduction Scene Holder – виконую генерацію початкової «сцени».

Схема переходів спливаючих вікон у застосуванні

Реалізовано приємну на вигляд вхідну сторінку(рисунок 5.1), з приємними м’якими кольорами. Це дозволяє зацікавити користувача відразу, що є важливою частиною розробки програмного забезпечення.

Рисунок 5.1 - Візуалізація головної сторінки

Після натиску кнопки «Start» ми переходимо на основну форму, на якій відбувається основна реалізація програми. Спочатку ми обираємо розмірність простору(3D чи 2D), натиснувши кнопку «2D» чи «3D» відповідно.

Рисунок 5.2 - Реакція програмного застосунку на натиск кнопки «Start»

Для 2 D

Як ми бачимо «сцена» відобразила, що вхідні дані вводяться для 2D моделі(рисунок 5.3).

Рисунок 5.3 - Реакція програмного застосунку на натиск кнопки «2D»

Після введення даних про кількість вершин та мінімальну і максимальну вартість переходу між вершинами натискаємо кнопку «Submit».

Рисунок 5.4 - Реакція програмного застосунку на введення вхідних даних

При натисканні «Generate field», ми бачимо, що на головному екрані зобразилась ліва крайня частина структури з відповідними координатами вершин в шестикутниках. З правого боку зобразились всі переходи між вершинами, що зображені на головній «сцені»(рисунок 5.5).

Рисунок 5.5 - Реакція програмного застосунку на натиск кнопки «Submit»

Після введення координат початкової та кінцевої кімнат для пошуку оптимального та всіх можливих шляхів натискаємо кнопку «Submit».

Після натиску кнопки «Number of ways» ми бачимо кількість всіх можливих ациклічних шляхів. В комірці кінцевої кімнати зображено відповідь, в даному випадку це 48649 шляхів(рисунок 5.7).

Після натиску кнопки «Dijkstra» бачимо відповідний оптимальний шлях від точки А до точки В знайдений алгоритмом Дейкстри(рисунок 5.8).

Після натиску кнопки «Ant» бачимо відповідний шлях від точки А до точки В знайдений мурашиним алгоритмом(рисунок 5.9).

Рисунок 5.6 - Реакція програмного застосунку на натиск кнопки «Generane field»

Рисунок 5.7 - Реакція програмного застосунку на натиск кнопки «Number of ways»

Рисунок 5.8 - Реакція програмного застосунку на натиск кнопки «Dijkstra»

Рисунок 5.9 - Реакція програмного застосунку на натиск кнопки «Ant»

Після натиску кнопок, які знаходяться під написом «Navigator» користувач рухається по будинку відповідно до натиснутої кнопки, при цьому після кожного натиску оновлюються список стінок(рисунок 5.10).

Рисунок 5.10 - Реакція програмного застосунку на натиск однієї із кнопок «Navigator»

Для 3D

Ми можемо бачити, що в нас вікно відобразило, що вхідні дані вводяться для 3D моделі(рисунок 5.11).

Рисунок 5.11 - Реакція програмного застосунку на натиск кнопки «3D»

Після введення даних про кількість вершин та мінімальну і максимальну ширину переходу між вершинами структури натискаємо кнопку «Submit»(рисунок 5.12).

Рисунок 5.12 - Реакція програмного застосунку на введення вхідних даних на натиск на кнопку «Submit»

При натисканні «Generate field», ми бачимо, що на головному екрані зобразились початкові шари будівлі з відповідними координатами структури в шарі. З правого боку зобразились всі шари, що зображені на головному екрані(рисунок 5.13).

Після введення координат початкової та кінцевої кімнат для пошуку оптимального та всіх можливих шляхів натискаємо кнопку «Submit». У вікні з’явиться відповідне повідомлення про підтвердження введення координат(рисунок 5.14).

Після натиску кнопки «Number of ways» ми бачимо кількість всіх можливих ациклічних шляхів. В комірці кінцевої координати зображено відповідь, в даному випадку це 1 шлях(рисунок 5.15).

Рисунок 5.13 - Реакція програмного застосунку на натиск кнопки «Generate field»

Рисунок 5.14 - Реакція програмного застосунку на введення координат початкової та кінцевої координати структури та натиск кнопки «Submit»

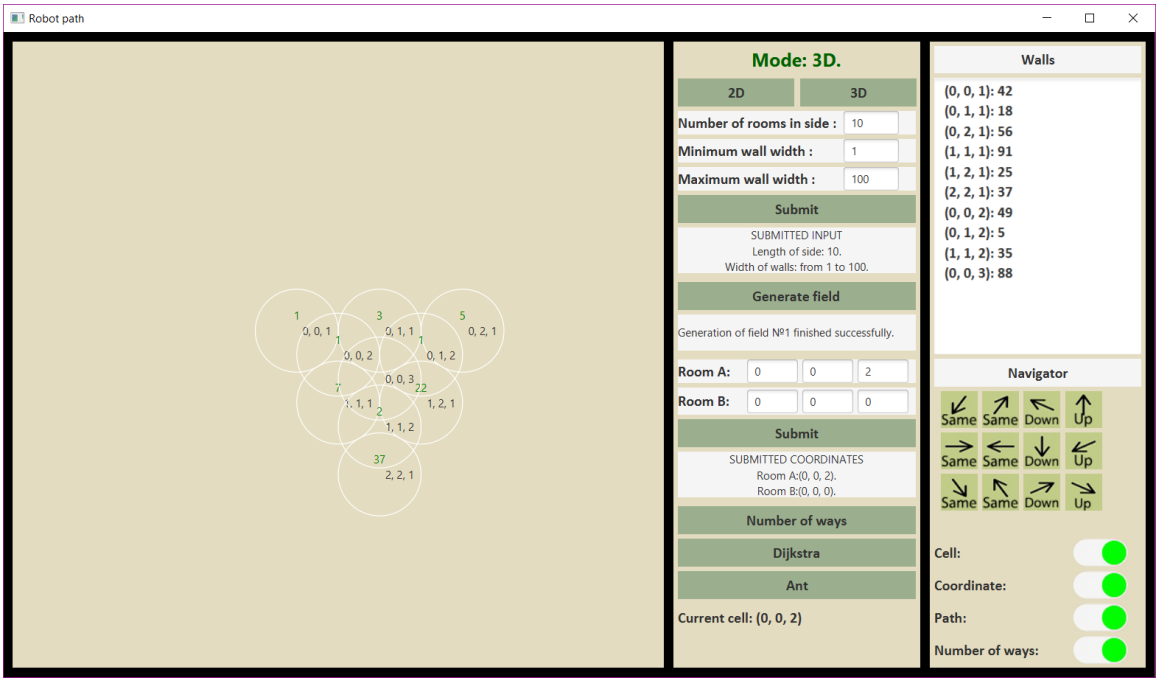


Рисунок 5.15 - Реакція програмного застосунку на натиск кнопки «Number of ways»

Після натиску кнопки «Dijkstra» бачимо відповідний оптимальний шлях від точки А до точки В знайдений алгоритмом Дейкстри(рисунок 5.16).

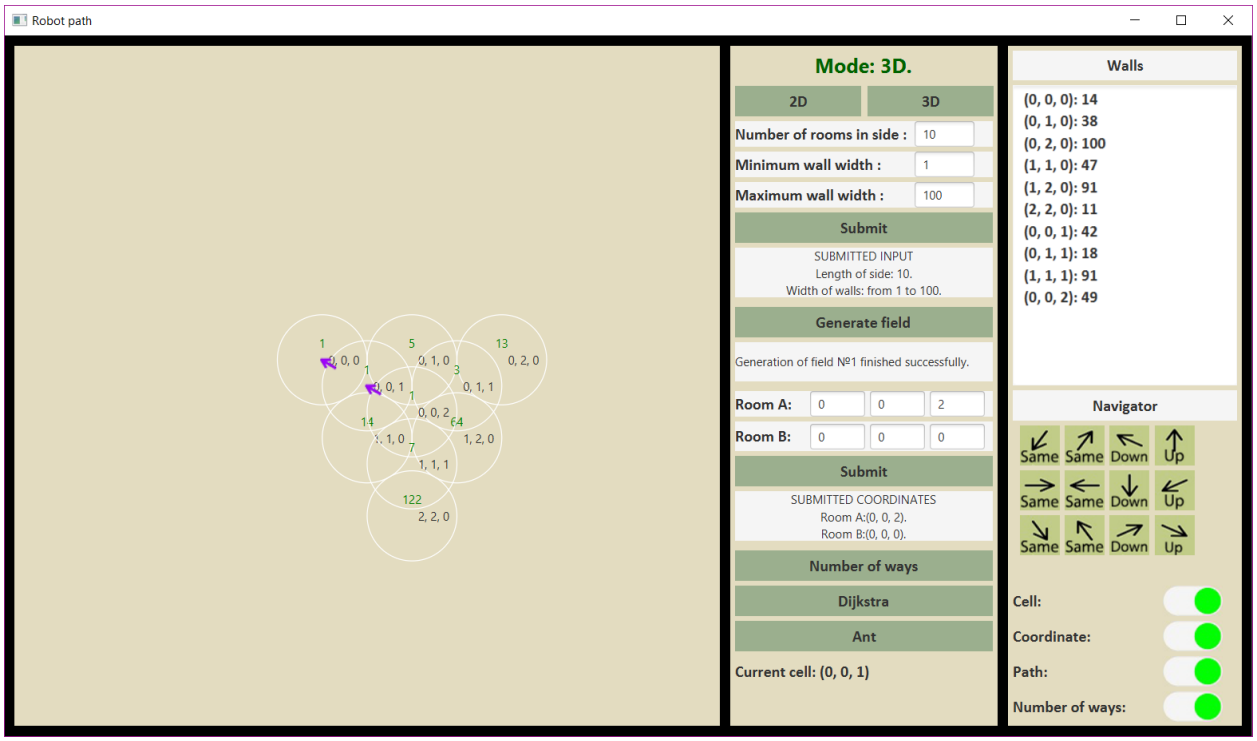


Рисунок 5.16 - Реакція програмного застосунку на натиск кнопки «Dijkstra»

Після натиску кнопки «Ant» бачимо відповідний шлях від точки А до точки В знайдений мурашиним алгоритмом(рисунок 5.17).

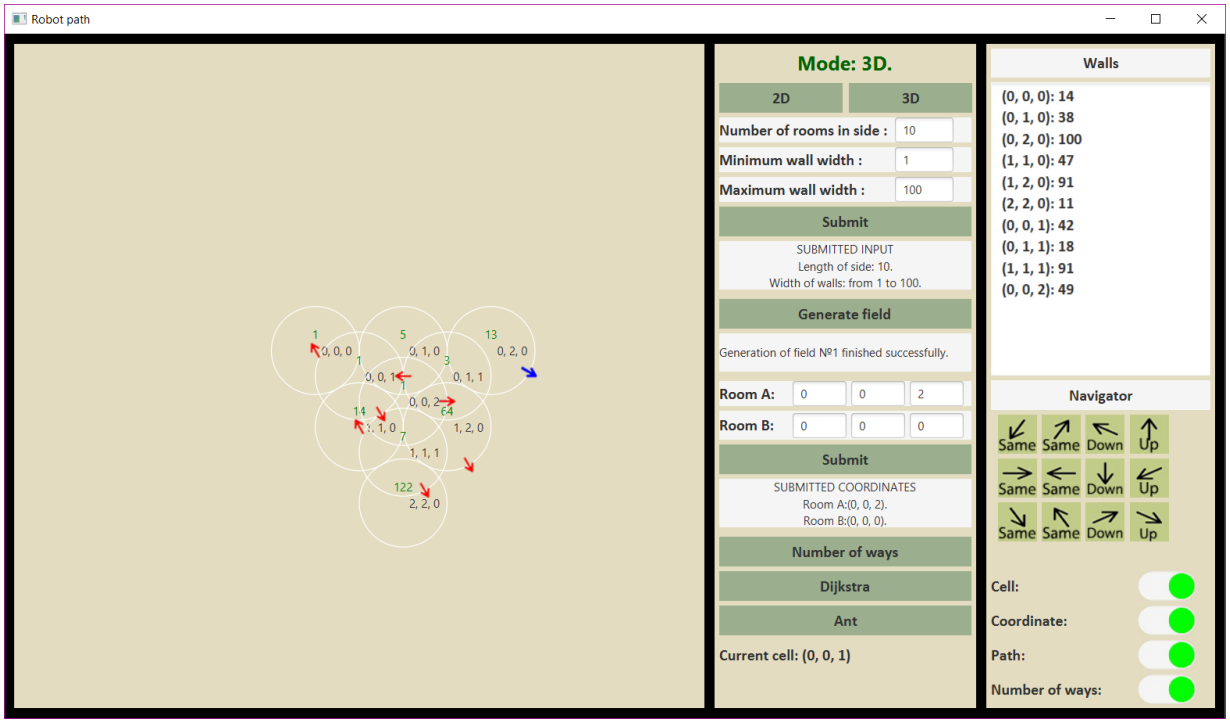


Рисунок 5.17 - Реакція програмного застосунку на натиск кнопки «Ant»

Після натиску кнопок, які знаходяться під написом «Navigation» користувач рухається по структурі відповідно до натиснутої кнопки, при цьому після кожного натиску оновлюються список переходів між вершинами(рисунок 5.18)

Реакції програмного продукту на командні перемикачі, які вмикають та вимикають відображення вершин структури, відображення координат, відображення знайденого системою оптимального шляху та відображення кількості взагалі можливих шляхів до заданої координати(рисунки 5.19 - 5.22).

Рисунок 5.18 - Реакція програмного застосунку на натиск кнопок «Naviganion»

Рисунок 5.19 - Реакція програмного застосунку на натиск перемикача «Path»

Рисунок 5.20 - Реакція програмного застосунку на натиск перемикача «Coordinate»

Рисунок 5.21 - Реакція програмного застосунку на натиск перемикача «Path»

Рисунок 5.22 - Реакція програмного застосунку на натиск перемикача «Number of ways»

Схема структурну переходу між вікнами наведено в графічних матеріалах.

5.2. Випробування програмного продукту

Основні моменти, що підлягають тестуванню – це своєчасна заповненість вхідних даних, коректна робота обробників подій, коректна обробка виключних ситуацій.

Конкретизуючи все перечислене ми будемо проводити тестування:

- отримання відповідних повідомлень і правильну обробку ситуацій при яких користувач не ввівши дані хоче отримати результат.
- можливість змінити вимірність системи після або до введення даних.
- можливість за допомогою навігації перетнути межі сформованої структури.
- коректної роботу графічних перемикачів.

5.2.1. Мета випробувань

Метою випробування є перевірка відповідності створеного функціонала відповідно до технічних вимог, а також перевірка коректної відповіді системи на непередбачені стандартами випадки.

5.2.2. Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3. Результати випробувань

В процесі тестування були перевірена уся функціональність системи комплексу задач. У наступних таблицях наведений перелік випробувань основних функціональних тестувань(таблиці 5.1 – 5.13).

Таблиця 5.1 – Початок роботи з програмою

Мета тесту:	**Перевірка функції «Вхід до налаштувань конфігурації робота»**
Початковий стан ПЗ	Відкрита початкова «сцена» для старту
Вхідні данні:	
Схема проведення тесту:	Натиснути кнопку «Start»
Очікуваний результат:	Відкрита головна «сцена»

Продовження таблиці 5.1

Стан ПЗ після проведення випробувань:	Відкрита головна «сцена»

Таблиця 5.2 – Вибір вимірності в якій буде проектуватись рух робота

Мета тесту:	**Перевірка функції «Вибір вимірності структури»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснути кнопку «2D» чи «3D»
Очікуваний результат:	З’явиться надпис «2D» чи «3D» над кнопками відповідно до обраного варіанту
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.3 – Перевірка заповнюваності вартості переходу між вершинами структури

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан ПЗ	Відкрита головна «сцена»

Продовження таблиці 5.3

Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести розмірність структури та не вказати вартості переходів між вершинами структури та натиснути на кнопку «Sumbit»
Очікуваний результат:	З’явиться попереджувальний надпис «Fields must contain only numbers»
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.4 – Перевірка заповнюваності розмірності структури

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести вартості переходів між вершинами структури та не вказати розмірність структури та натиснути на кнопку «Sumbit»
Очікуваний результат:	З’явиться попереджувальний надпис «Fields must contain only numbers»
Стан ПЗ після проведення	Відкрита основна «сцена»

Мета тесту:	Перевірка функції «Перевірка заповнюваності»
випробувань:	

Таблиця 5.5 – Перевірка отримання вхідних даних системою

Мета тесту:	Перевірка функції «Перевірка заповнюваності»
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести вартості переходів між вершинами структури та вказати розмірність структури та натиснути на кнопку «Sumbit»
Очікуваний результат:	З’явиться попереджувальний надпис «SUMBMITTED INPUT»
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.6 – Перевірка візуально зображення в результаті генерації структури

Мета тесту:	Перевірка функції «Генерація та зображення структури»
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	

Продовження таблиці 5.6

Схема проведення тесту:	Натиснути на кнопку «Generate field»
Очікуваний результат:	З’явиться візуальне зображення північно-західної частини структури
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.7 – Перевірка отримання координат пошуку системою

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	Координати кімнати А та координати кімнати В
Схема проведення тесту:	Отримання від користувача початкової та кінцевої координат вершин структури та натиснення кнопки «Submit»
Очікуваний результат:	З’явиться повідомлення «SUBMITTED COORDINATES»
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.8 – Перевірка введення неправильних координат пошуку системою

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	Координати кімнати A та координати кімнати B
Схема проведення тесту:	Отримання від користувача початкової та кінцевої координат вершин структури невідповідного типу та натиснення кнопки «Submit»
Очікуваний результат:	З’явиться повідомлення «FIELDS MUST CONTAIN ONLY NUMBERS»
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.9 – Перевірка коректності пошуку кількості взагалі можливих шляхів

Мета тесту:	**Перевірка функції «Пошук кількості шляхів»**
Початковий стан ПЗ	Відкрита головна «сцена»

Продовження таблиці 5.9

Вхідні данні:	
Схема проведення тесту:	Натиснення кнопки «Number of ways»
Очікуваний результат:	З’явиться в кожній координаті на візуальному зображені зеленим кольором кількість взагалі можливих шляхів
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.10 – Перевірка коректності пошуку оптимального шляху

Мета тесту:	**Перевірка функції «Пошук оптимального шляху»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення кнопки «Dijkstra» або «Ant»
Очікуваний результат:	З’являться на візуальному зображені різнокольорові стрілки між вершинами структури, що вказуватимуть оптимальний шлях

Продовження таблиці 5.10

Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.11 – Перевірка коректності роботи навігації

Мета тесту:	**Перевірка функції «Навігація»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення будь-якої стрілки із наведених для навігації стрілок
Очікуваний результат:	Візуальне зображення структури змінить фокус на одну із координат в залежності від обраної користувачем стрілки
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.12 – Перевірка коректності роботи перемикачів графічного відображення елементів структури

Мета тесту:	**Перевірка функції «Графічні перемикачі»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	

Продовження таблиці 5.12

Схема проведення тесту:	Натиснення будь-якого з перемикачів відображення графічних об’єктів
Очікуваний результат:	При ввімкнені відповідні графічні об’єкти з’являться, при вимкнені графічні об’єкти зникнуть
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 5.13 – Перевірка коректності роботи системи в випадку коли користувач навігацією перетне крайні межі структури

Мета тесту:	**Перевірка функції «Навігація»**
Початковий стан ПЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натискання користувачем однієї і тієї ж кнопки навігації до тих пір поки візуальне зображення структури не перетне межі структури
Очікуваний результат:	Відображення пустого зображення структури
Стан ПЗ після проведення випробувань:	Відкрита основна «сцена»

Висновок до розділу

В даному розділі описано основний функціонал системи з покроковим керівництвом використовуючи скріншоти обробників подій на «сцені». В розділі обумовлено основні актори сцени та їх функціонал.

В підрозділі «Випробування програмного продукту» було наведено тести, які покрили весь функціонал системи.

ЗАГАЛЬНІ ВИСНОВКИ

В даній представлено вирішення надзвичайно важливої на актуальний час, а саме якнайшвидшу та безперечно найбільш безпечну допомогу рятувальникам під час рятувальних дій після катастрофи, яка несе в собі обвали або інші рушійні дії, в наслідок чого виживання людей, що потрапили в катастрофу, знаходиться під питанням.

Створений програмний продукт дозволить дистанційно налаштувати робота рятівника після чого останній шляхом аналізу структури, де сталась катастрофа, отримає змодельований шлях, що дозволить якнайшвидше використати такі важливі для порятунку хвилини пошуку.

ПЕРЕЛІК ПОСИЛАНЬ

1) 5 Robots That May Rescue You From Natural Disasters [Електронний ресурс]//Режим доступу: https://www.govtech.com/em/safety/5-Robots-That-May-Rescue-You-From-Natural-Disasters.html.

2) BROKK. Техника, машины, роботы [Електронний ресурс]//]//Режим доступу: http://www.brokk.ru/produktsiya/brokk-500/.

3) Construction - Brokk Global [Електронний ресурс]//Режим доступу: https://www.brokk.com/industry/construction/.

4) Rescue robots into action [Електронний ресурс]//Режим доступу: https://uk.wikipedia.org/wiki/Терористичний_акт_11_вересня_2001_року.

5) Pros and cons of Java programming [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/nuances-of-programming/плюсы-и-минусы-программирования-на-java-2861f4c2a0d5.

6) Ant colony optimization algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms.

7) Dijkstra's algorithm and its implementation by means of STL – Режим доступу до ресурсу: https://www.e-olymp.com/uk/blogs/posts/21.

8) JavaFX [Електронний ресурс] – Режим доступу до ресурсу: https://openjfx.io.

9) JavaFX [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JavaFX.

Додаток А

Тексти програмного коду

Robot Path

(Найменування програми (документа))

DVD-R

(Вид носія даних)

12 арк, 2,21 Мб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Class Algorithm2D

```
package algorithm;
public class Algorithm2D {
   private HexagonAlgorithmField field;
   private int lengthSide;
   private int startY;
   private int startX;
   private boolean isExist(int x, int y) {
      return (Math.abs(x - y) <= lengthSide - 1) && (x < 2 * lengthSide - 1) && (y < 2 * lengthSide - 1);
   }
   public Algorithm2D(HexagonAlgorithmField field) {
      this.field = field;
      this.lengthSide = field.getLengthOfSide();
   }
   long calculateNumberWays(int x, int y) {
      AlgorithmCell target = field.getCell(x, y);
      AlgorithmCell topNeighbour =  field.getCell(x, y - 1);
      AlgorithmCell leftTopNeighbour =field.getCell(x - 1, y - 1);
      AlgorithmCell leftBottomNeighbour = field.getCell(x - 1, y);
      long result = 0;
      if (topNeighbour != null) {
         result += topNeighbour.getNumberOfWays();
      }
      if (leftTopNeighbour != null) {
         result += leftTopNeighbour.getNumberOfWays();
      }
      if (leftBottomNeighbour != null) {
         result += leftBottomNeighbour.getNumberOfWays();
      }
         target.setNumberOfWays(result);
      return result;
   }
   public long calculateNumberWays(int startX, int startY, int endX, int endY) {
      field.getCell(startX, startY).setNumberOfWays(1);
      this.startX = startX;
```

```
        this.startY = startY;
        int numberOfLevels = Math.min((endX - startX), (endY - startY));

        for (int i = 0; i <= numberOfLevels; i++) {            int currentX = startX + i;
            int currentY = startY + i;
            while (isExist(currentX, currentY)) {     if ((currentX > startX) || (currentY > startY))
                    calculateNumberWays(currentX, currentY);
                currentY++;
            }           currentX = startX + i;
            currentY = startY + i;
            while (isExist(currentX, currentY)) {
                if ((currentX > startX) || (currentY > startY))
                    calculateNumberWays(currentX, currentY);
                currentX++;
            }
        }
        return field.getCell(endX, endY).getNumberOfWays();
    }
    public static void main(String[] args) {
        HexagonAlgorithmField hexagonAlgorithmField = new HexagonAlgorithmField(4);
        System.out.println(new Algorithm2D(hexagonAlgorithmField).calculateNumberWays(1, 1, 6, 6));
        System.out.println();
    }
}
```

Class Algorithm3D

```
package algorithm;
public class Algorithm3D {
    private SphereAlgorithmField field;
    private int lengthSide;
    private int startX;
    private int startY;
    private int startZ;
    private boolean isExist(int x, int y, int z) {
        int row = lengthSide - 1 - y;
        if ((x >= 0) && (x < lengthSide) &&
                (row >= 0) && (row < lengthSide - x) &&
```

```
            (z >= 0) && (z < row + 1)) {
        return true;
    }
    return false;
}
public Algorithm3D(SphereAlgorithmField field) {
    this.field = field;
    this.lengthSide = field.getLengthOfSide();
}
int calculateNumberWays(int x, int y, int z) {
    AlgorithmCubicle target = field.getCell(x, y, z);
    AlgorithmCubicle xy_1z_1 = field.getCell(x, y - 1, z + 1);
    AlgorithmCubicle xyz_1 = field.getCell(x, y, z + 1);
    AlgorithmCubicle x_1y_1z_1 = field.getCell(x - 1, y - 1, z + 1);
    AlgorithmCubicle xy_1z = field.getCell(x, y - 1, z);
    AlgorithmCubicle x_1y_1z = field.getCell(x - 1, y - 1, z);
    AlgorithmCubicle x_1yz = field.getCell(x - 1, y, z);
    int result = 0;
    if (xy_1z_1 != null && xy_1z_1.getNumberOfWays() > -1) {
        result += xy_1z_1.getNumberOfWays();
    }
    if (xyz_1 != null && xyz_1.getNumberOfWays() > -1) {
        result += xyz_1.getNumberOfWays();
    }
    if (x_1y_1z_1 != null && x_1y_1z_1.getNumberOfWays() > -1) {
        result += x_1y_1z_1.getNumberOfWays();
    }
    if (xy_1z != null && xy_1z.getNumberOfWays() > -1) {
        result += xy_1z.getNumberOfWays();
    }
    if (x_1y_1z != null && x_1y_1z.getNumberOfWays() > -1) {
        result += x_1y_1z.getNumberOfWays();
    }
    if (x_1yz != null && x_1yz.getNumberOfWays() > -1) {
        result += x_1yz.getNumberOfWays();
    }
```

```
        if (result > 9_9999_999 ||
                (xy_1z_1 != null && xy_1z_1.getNumberOfWays() == 0 && isExist(x, y - 1, z + 1))
                || (xyz_1 != null && xyz_1.getNumberOfWays() == 0 && isExist(x, y, z + 1))
                || (x_1y_1z_1 != null && x_1y_1z_1.getNumberOfWays() == 0 && isExist(x - 1, y - 1, z + 1))
                || (xy_1z != null && xy_1z.getNumberOfWays() == 0 && isExist(x, y - 1, z))
                || (x_1y_1z != null && x_1y_1z.getNumberOfWays() == 0 && isExist(x - 1, y - 1, z))
                || (x_1yz != null && x_1yz.getNumberOfWays() == 0 && isExist(x - 1, y, z))) {
            target.setNumberOfWays(0);
        } else {
            target.setNumberOfWays(result);
        }
        return result;
    }
    public int calculateNumberWays(int startX, int startY, int startZ, int endX, int endY, int endZ) {
        field.getCell(startX, startY, startZ).setNumberOfWays(1);
        this.startX = startX;
        this.startY = startY;
        this.startZ = startZ;
        for (int level = startZ; level >= endZ ; level--) {
            int currentRowBeginningX = startX;
            int currentRowBeginningY = startY;
            while(true){
                if (!isExist(currentRowBeginningX,currentRowBeginningY,level)){
                    currentRowBeginningY = currentRowBeginningX;
                }
                int currentY = currentRowBeginningY;
                while(isExist(currentRowBeginningX, currentY, level)){
                    if(!((currentRowBeginningX == startX) && (currentY == startY) &&(level == startZ)))
                        calculateNumberWays(currentRowBeginningX, currentY, level);
                    currentY++;
                }
                currentRowBeginningX++;
                if(currentRowBeginningX > lengthSide - level - 1){
                    break;
                }
    //              currentRowBeginningY++;
```

```
            }
        }

        return field.getCell(endX, endY, endZ).getNumberOfWays();
    }
}
```

Class AlgorithmCell

```
package algorithm;
}

public class AlgorithmCell extends Cell {
    private long numberOfWays;
    public AlgorithmCell(int x, int y) {
        super(x, y);   }
    public long getNumberOfWays() {
        return numberOfWays;    }
    public void setNumberOfWays(long numberOfWays) {
        this.numberOfWays = numberOfWays;  }
```

class AlgorithmCubicle

```
package algorithm;
public class AlgorithmCubicle {
    private int x;
    private int y;
    private int z;
    private int numberOfWays;
    public AlgorithmCubicle(int x, int y, int z) {
        this.x = x;
        this.y = y;
       this.z = z;    }
    public AlgorithmCubicle(int x, int y, int z, int numberOfWays) {
        this(x,y,z);
        this.numberOfWays = numberOfWays;
    }
    public int getX(){
        return x;
    }
```

```
    public int getY() {
        return y;
    }
    public int getZ() {
        return z;
    }
    public int getNumberOfWays() {
        return numberOfWays;

    public void setNumberOfWays(int numberOfWays) {
        this.numberOfWays = numberOfWays;    }}
```

Class Cell

```
package algorithm;
public class Cell {
    private int x,y;
    public Cell(int x,int y){
        this.x = x;
        this.y = y;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public void setX(int x) {
        this.x = x;
    }
    public void setY(int y) {
        this.y = y;
    }
}
```

HexagonAlgorithmField

```
package algorithm;
import java.util.ArrayList;
```

```
public class HexagonAlgorithmField {
    private ArrayList<AlgorithmCell> cells = new ArrayList<>();
    private int lengthOfSide;
    public HexagonAlgorithmField(int n) {
        lengthOfSide = n;
        for (int i = 0; i < (2 * n - 1); i++) {
            //constructing main line
            cells.add(new AlgorithmCell(i, i));
        }
        for (int i = 0; i < (n - 1); i++) {
            //constructing top part
            for (int j = 0; j < (2 * n - 2 - i); j++) {
                cells.add(new AlgorithmCell(i+j+1, j));}
            //constructing bottom part
            for (int j = 0; j < (2 * n - 2 - i); j++) {
                cells.add(new AlgorithmCell(j, i+j+1));  }}}
    //return null if cell has invalid coordinates
    public AlgorithmCell getCell(int x, int y) {
        AlgorithmCell result = null;
        for (AlgorithmCell cell : cells) {
            if ((cell.getX() == x) && (cell.getY()) == y) {
                result = cell;}  }
        return result; }
    public int getLengthOfSide() {
        return lengthOfSide;
    }
    public void setLengthOfSide(int lengthOfSide) {
        this.lengthOfSide = lengthOfSide;
    }
}
```

Class SphereAlgorithmField

```
package algorithm;
import java.util.ArrayList;
public class SphereAlgorithmField {
    private ArrayList<AlgorithmCubicle> cells = new ArrayList<>();
    private int lengthOfSide;
```

```
    public SphereAlgorithmField(int n){
        lengthOfSide = n;
        for (int index = 0; index < lengthOfSide; index++) {
            for (int row = 0; row < lengthOfSide - index ; row++) {
                for (int level = 0; level < row+1; level++) {
                    cells.add (new AlgorithmCubicle (index, lengthOfSide - 1 - row,level, -1));
                }
            }
        }
    }
    public AlgorithmCubicle getCell(int x, int y, int z){
        AlgorithmCubicle result = null;
        for (AlgorithmCubicle cell : cells) {
            if((cell.getX() == x) && (cell.getY() == y)
                && (cell.getZ() == z)){
                result = cell;
            }
        }
        return result;
    }

    public int getLengthOfSide(){
        return lengthOfSide;
    }

    public void setLengthOfSide(int lengthOfSide) {
        this.lengthOfSide = lengthOfSide;
    }
}
```

AntAlgoritm

```
package algorithmMinPath;

import java.util.concurrent.ThreadLocalRandom;
```

```
public class AntAlgoritm {
    private int startPoint;
    private int endPoint;
    private int[] ofVisit;
    private int[][] matrixOfShortestArcs;
    private double sumEta;
    private int lenthSide;
    private double[] arrayOfPheromones;
    private double[] probabilityArray;
    private double[] etaForTheCurrentPoint;
    private int[] finishArrayOfPathWays;
    private int pathLength;

    private int pseudoRand(double count) {
        return ThreadLocalRandom.current().nextInt(0, ((int) count) + 1);
    }

    public AntAlgoritm(int lenthSide, int startPoint, int endPoint, int[][] matrixOfShortestArcs) {
        this.lenthSide = lenthSide;
        this.startPoint = startPoint;
        this.endPoint = endPoint;
        this.matrixOfShortestArcs = matrixOfShortestArcs;
        ofVisit = new int[matrixOfShortestArcs.length];
    }

    public void start() {
        fillingArrayOfProbability(startPoint);
    }

    public void initializationArrayOfProbability() {

        for (int counter = 0; counter < matrixOfShortestArcs.length; counter++) {
            if ((etaForTheCurrentPoint[counter] != -1) && (ofVisit[counter] != 1)) {
                probabilityArray[counter] = (etaForTheCurrentPoint[counter] * arrayOfPheromones[counter]) / sumEta;
                helpToFillProbilityArray(counter);
            }
```

```
    }
}

public void refreshArrayOfPheromones() {
    for (int counter = 0; counter < finishArrayOfPathWays.length; counter++) {
        if (finishArrayOfPathWays[counter] > 0) {
            if (arrayOfPheromones[finishArrayOfPathWays[counter]] > lenthSide / 7)
                arrayOfPheromones[finishArrayOfPathWays[counter]] -= lenthSide / 7;
        }
    }
}

public int fillingArrayOfProbability(int numberOfVertex) {
    pathLength = 0;
    for (int counter = 0; counter < matrixOfShortestArcs.length; counter++) {
        ofVisit[counter] = 0;
    }
    ofVisit[startPoint] = 1;
    finishArrayOfPathWays = new int[matrixOfShortestArcs.length];
    for (int counter = 0; counter < finishArrayOfPathWays.length; counter++) {
        finishArrayOfPathWays[counter] = -1;
    }
    int counter = 0;
    do {
        probabilityArray = new double[matrixOfShortestArcs.length];
        fillingArrayOfEta(numberOfVertex);
        initializationArrayOfProbability();
        finishArrayOfPathWays[counter] = numberOfVertex;
        numberOfVertex = changeNextVertex();
        if (counter > 0) {
        }
        if (numberOfVertex == -1) {
            refreshArrayOfPheromones();
            fillingArrayOfProbability(startPoint);
            break;
        }
```

```
            counter++;
        } while (numberOfVertex != endPoint);
        System.out.print("");
        return 0;
    }

    private void helpToFillProbilityArray(int numberOfVertex) {
        for (int counter = numberOfVertex - 1; counter >= 0; counter--) {
            if ((probabilityArray[counter] != 0) && (ofVisit[counter] != 1)) {
                probabilityArray[numberOfVertex] += probabilityArray[counter];
                break;
            }
        }
    }

    public void fillingArrayOfEta(int numberOfVertex) {
        sumEta = 0;
        arrayOfPheromones = new double[matrixOfShortestArcs.length];
        etaForTheCurrentPoint = new double[matrixOfShortestArcs.length];
        for (int counter = 0; counter < etaForTheCurrentPoint.length; counter++) {
            etaForTheCurrentPoint[counter] = -1;
            arrayOfPheromones[counter] = lenthSide;
        }
        for (int counter = 0; counter < etaForTheCurrentPoint.length; counter++) {
            if ((matrixOfShortestArcs[numberOfVertex][counter] != Integer.MAX_VALUE) && (counter != numberOfVertex)) {
                etaForTheCurrentPoint[counter] = (1 / (double) matrixOfShortestArcs[numberOfVertex][counter]);
                sumEta += etaForTheCurrentPoint[counter] * arrayOfPheromones[counter];
            }
        }
    }

    public int changeNextVertex() {
        double maxElement = -1;
        for (int n = 0; n < probabilityArray.length; n++) {
            if (probabilityArray[n] > maxElement)
                maxElement = probabilityArray[n];
```

```
        }
        int randomNumber = pseudoRand(maxElement * 100);
        return checkRandomNumberInProbabilitiArray(randomNumber);
    }

    public int checkRandomNumberInProbabilitiArray(int randomNumber) {
        for (int counter = probabilityArray.length - 1; counter >= 0; counter--) {
            if ((probabilityArray[counter] > 0) && (ofVisit[counter] != 1)) {
                for (int secondCounter = counter; secondCounter >= 0; secondCounter--) {
                    if ((secondCounter == 0) && (ofVisit[counter] != 1)) {
                        if ((0 <= randomNumber) && (probabilityArray[counter] * 100 >= randomNumber)) {
                            ofVisit[counter] = 1;
                            return counter;
                        }
                    }
                    if ((probabilityArray[secondCounter] > 0) && (ofVisit[counter] != 1) && (counter != secondCounter)) {
                        if ((probabilityArray[secondCounter] * 100 <= randomNumber) && (probabilityArray[counter] * 100 >=
randomNumber)) {
                            ofVisit[counter] = 1;
                            return counter;
                        } else
                            break;
                    }
                }
            }
        }
        return -1;
    }

    public String answer2D(PreparationForDijkstra2D preparationForDijkstra2D) {
        for (int counter = 0; counter < finishArrayOfPathWays.length; counter++) {
            if (finishArrayOfPathWays[counter] == -1) {
                finishArrayOfPathWays[counter] = endPoint;
                break;
            }
        }
        String result = null;
```

```
        result = "(";
        int counter = 0;
        do {
            result += preparationForDijkstra2D.findPoint(finishArrayOfPathWays[counter]) + ") (";
            counter++;
        } while (finishArrayOfPathWays[counter] != endPoint);
        findPathLenth();
        result += preparationForDijkstra2D.findPoint(finishArrayOfPathWays[counter]) + "): " + pathLength;
//        System.out.print(result);
        return result;
    }

    public String answer3D(PreparationForDijkstra3D preparationForDijkstra3D) {
        for (int counter = 0; counter < finishArrayOfPathWays.length; counter++) {
            if (finishArrayOfPathWays[counter] == -1) {
                finishArrayOfPathWays[counter] = endPoint;
                break;
            }
        }
        String result = null;
        result = "(";
        int counter = 0;
        do {
            result += preparationForDijkstra3D.toPoint(finishArrayOfPathWays[counter]) + ") (";
            counter++;
        } while (finishArrayOfPathWays[counter] != endPoint);
        findPathLenth();
        result += preparationForDijkstra3D.toPoint(finishArrayOfPathWays[counter]) + "): " + pathLength;
//        System.out.print(result);
        return result;
    }
        }
    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проекту

_____ *М.О. Солдатова*

(підпис) (ініціали, прізвище)

“17” квітня 2019 р.

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ *О.А. Павлов*

(підпис) (ініціали, прізвище)

“18” квітня 2019 р.

Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5221.1180-с.ТЗ*

На 10 сторінках

Київ – 2019 року

ЗМІСТ

Відтворювати

Зм.	Арк.	Підпис	Дата				
Розроб.	Поліщук А.О.			Автоматизація пошуку оптимального шляху робота-рятівника в умовах невизначеної структури замкнутого робочого простору	Літ.	Лист	Листів
Перевірив.	Солдатова М.О.					2	10
Н. кон.	Халус О.А.				КПІ ім. ІгоряСікорського кафедра АСОІУ гр. ІС-52		
Затв.	Павлов О.А.						

					ДП ІС-5221.1181-с.ТЗ		
Зм.	Арк.	Прізвище	Підпис	Дата			
Розроб.		Поліщук А.О.			Автоматизація пошуку оптимального шляху робота-рятівника в умовах невизначеної структури замкнутого робочого простору	Літ.	Лист 2 / Листів 10
Перевірив.		Солдатова М.О.					
Н. кон.		Халус О.А.				КПІ ім. ІгоряСікорського кафедра АСОІУ гр. ІС-52	
Затв.		Павлов О.А.					

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: *Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору.*

1.2 Найменування організації-замовника та організації-учасника робіт

Генеральним замовником проекту являється ТОВ «Бізнес Сервіс Провайдер». Представником замовника є Халтін Юрій Олександрович.

Розробником системи є студент групи ІС-52 факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського» Поліщук Андрій Олегович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням автоматизованого пошуку оптимального шляху робота-рятівника – 5 лютого 2018 рік.

Плановий термін по закінченню роботи над створенням автоматизованого пошуку оптимального шляху робота-рятівника – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення комплексу задач

Призначенням розробки є створення абсолютно автономного робота, який в небезпечній для життя ситуації допоможе якнайшвидше провести рятувальні дії.

2.2 Цілі створення комплексу задач

Цілями розробки є автоматизація проведення рятувальних пошукових робіт за найкращий, враховуючи всі умови, час за рахунок:

- Урахування в системі в якості параметра пошуку не тільки час, а й інші параметри;
- Алгоритм пошуку варіюється в залежності від вхідних даних;
- Користувач, при необхідності, може сам обрати алгоритм пошуку.

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

- реалізувати алгоритми пошуку оптимального шляху робота-рятівника;
- розробити систему вибору алгоритму пошуку оптимального шляху покладаючись на отриману інформації вхідних даних;
- розробити підсистему інтерфейсу, що змоделює для остаточного ухвалення план руху робота;
- розробити підсистему автоматичної генерації початкових умов для подальшого тестування застосунку клієнтами;
- розробити підсистему, яка буде працювати як і в двовимірному так і в тривимірному просторі(в перспективі розвитку робототехніки).

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для розробки програмного продукту було обрано фреймворк розробки настільних застосунків JavaFX з використанням найпоширенішим архітектурним шаблоном для даної платформи MVVM.

Так, шаблон архітектури Model-View-ViewModel (MVVM) розділяє продукт на три основні компоненти: модель, представлення та логіку представлення.

MVVM являє собою стандартний шаблон розробки, який походить від шаблонів MVP та MVC

Для написання коду було обрано мову Java. Застосунок може працювати на будь-якому, де присутня JVM.

Користувач може задавати власні налаштування генерації та передавати вхідні дані моделі.

Об'єктом автоматизації є пошук оптимального шляху роботом-рятівником під час рятувальних операцій за допомогою JavaFX.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Даний програмний продукт має генерувати інформаційні структури згідно заданих налаштувань за допомогою яких буде моделюватись оптимальний шлях та виконувати наступні функції:

1. Система повинна надати користувачу можливість вводити налаштування генерації;
2. Система повинна надати користувачу можливість обрати між автоматичним генеруванням структур даних чи читанням їх з файлу;
3. Система повинна надавати користувачу можливість змінювати вхідні параметри не формуючи ще раз інформаційні структури.
   - 3.1. Система повинна надавати можливість користувачу змінювати довжину стінки будівлі;
   - 3.2. Система повинна надавати можливість користувачу змінювати максимальне та мінімальне обмеження для ширини стінок;
   - 3.3. Система повинна надавати можливість користувачу змінювати алгоритм пошуку та можливість надати системі автоматично визначити алгоритм;
   - 3.4. Система повинна надавати можливість користувачу змінювати напрям пошуку робота-рятівника.

4.2 Вимоги до надійності

Система повинна містити обробку введення користувачем некоректних даних:

- Система повинна видавати повідомлення, коли користувач вводить некоректні дані.
- Система повинна надавати можливість ввести інші вхідні дані, коли користувач вводить некоректні дані.

Система повинна бути працездатною після зміни алгоритму пошуку чи будь-яких інших параметрів.

4.3 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити процедуру генерації зруйнованої будівлі вказаного розміру та складності.

Для правильної роботи розробленої системи до складу технічних засобів потрібен комп’ютор з наступними вимогами:

- конфігурація комп’ютора:

1) двох ядерний процесор з частотою не нижче 2.5 ГГц;

2) достатній об’єм оперативної пам’яті (не менше 4 ГБ);

3) відеокарта з об’ємом пам’яті не менше 2048 Мб та частотою графічного процесора 1006 МГц.

- програмне забезпечення:

Необхідним обладнанням для роботи системи є комп’ютер зі встановленим середовищем Java Runtime Environment(jre), яка необхідна для виконання Java-застосунків, без компілятора та інших середовищ розробки та операційна система Windows 7, Windows 8 та Windows 10 64-bit або Mac OS X 10.9.2.

JRE містить в собі віртуальну машину – Java Virtual Machine і бібліотеки Java-класів.

- комп’ютерна периферія, до складу якої входить:

1) монітор;

2) комп’ютерна миша;

3) клавіатура.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з автоматизації формування асортименту торгівельної організації.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	07.02.2019	
2.	Розробка сценарію роботи	12.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	20.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	02.03.2019	
5.	Розробка інформаційного забезпечення	17.03.2019	
6.	Розробка програмного забезпечення	29.03.2019	
7.	Налагодження програми	13.04.2019	
8.	Тестування програми	27.04.2019	
9.	Здача готового програмного продукту замовнику	12.05.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Для контролю правильності роботи програмного забезпечення буде проведено модульне тестування (Unit Test). В ході тестування будуть перевірені всі граничні умови вхідних даних, будуть перевірені всі вищеобумовлені вимоги. Буде проведено випробування коректності роботи алгоритмів при автоматичній генерації даних та отримання даних з файлу.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *М.О. Солдатова*	_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“13” травня 2019 р.	“14” травня 2019 р.

Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5221.1180-с.ПМВ*

на 14 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Повне найменування системи: *Автоматизація пошуку оптимального шляху робота-рятівника в умовах невизначеної структури замкнутого робочого простору.*

1.2 Область застосування

Програма використовується для автоматизація рятувальних дій для забезпечення найбільш високої швидкодії рятувальних операцій, застосовуючи алгоритми пошуку оптимального шляху. Область застосування покриває всі рятувальні операції, що є небезпечними для життя, та існує наявний план структури місцевості на якій трапилась катастрофа.

1.3 Умовне позначення програми

Умовне позначення: «RobotPath».

2 МЕТА ВИПРОБУВАНЬ

Метою випробування є перевірка відповідності створеного функціонала відповідно до технічних вимог, а також перевірка коректної відповіді системи на непередбачені стандартами випадки.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Даний програмний продукт має генерувати інформаційні структури згідно заданих налаштувань за допомогою яких буде моделюватись оптимальний шлях та виконувати наступні функції:

1. Система повинна надати користувачу можливість вводити налаштування генерації;

2. Система повинна надати користувачу можливість обрати між автоматичним генеруванням структур даних чи читанням їх з файлу;

3. Система повинна надавати користувачу можливість змінювати вхідні параметри не формуючи ще раз інформаційні структури.

3.1. Система повинна надавати можливість користувачу змінювати довжину стінки будівлі;

3.2. Система повинна надавати можливість користувачу змінювати максимальне та мінімальне обмеження для ширини стінок;

3.3. Система повинна надавати можливість користувачу змінювати алгоритм пошуку та можливість надати системі автоматично визначити алгоритм;

3.4. Система повинна надавати можливість користувачу змінювати напрям пошуку робота-рятівника.

3.1.1 Вимоги до надійності

Система повинна містити обробку введення користувачем некоректних даних:

- Система повинна видавати повідомлення, коли користувач вводить некоректні дані.
- Система повинна надавати можливість ввести інші вхідні дані, коли користувач вводить некоректні дані.

Система повинна бути працездатною після зміни алгоритму пошуку чи будь-яких інших параметрів.

3.1.2 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити процедуру генерації зруйнованої будівлі вказаного розміру та складності.

Для правильної роботи розробленої системи до складу технічних засобів потрібен комп’ютор з наступними вимогами:

- конфігурація комп’ютора:

1) двох ядерний процесор з частотою не нижче 2.5 ГГц;

2) достатній об’єм оперативної пам’яті (не менше 4 ГБ);

3)) відеокарта з об’ємом пам’яті не менше 2048 Мб та частотою графічного процесора 1006 МГц.

- програмне забезпечення:

Необхідним обладнанням для роботи системи є комп’ютер зі встановленим середовищем Java Runtime Environment(jre), яка необхідна для виконання Java-застосунків, без компілятора та інших середовищ розробки та операційна система Windows 7, Windows 8 та Windows 10 64-bit або Mac OS X 10.9.2.

JRE містить в собі віртуальну машину – Java Virtual Machine і бібліотеки Java-класів.

- комп’ютерна периферія, до складу якої входить:

1) монітор;

2) комп’ютерна миша;

3) клавіатура.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація має складатися з керівництва користувача та вихідних текстів програмного коду.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Для контролю правильності роботи програмного забезпечення буде проведено модульне тестування (Unit Test). В ході тестування будуть перевірені всі граничні умови вхідних даних, будуть перевірені всі вищеобумовлені вимоги. Буде проведено випробування коректності роботи алгоритмів при автоматичній генерації даних та отримання даних з файлу.

6 МЕТОДИ ВИПРОБУВАНЬ

6.1 Функціональне тестування

В процесі тестування були перевірена уся функціональність системи комплексу задач. У наступних таблицях наведений перелік випробувань основних функціональних тестувань(таблиця 6.1 – 6.13).

Таблиця 6.1 – Початок роботи з програмою

Мета тесту:	**Перевірка функції «Вхід до налаштувань конфігурації робота»**
Початковий стан КЗ	Відкрита початкова «сцена» для старту
Вхідні данні:	
Схема проведення тесту:	Натиснути кнопку «Start»
Очікуваний результат:	Відкрита головна «сцена»
Стан КЗ після проведення випробувань:	Відкрита головна «сцена»

Таблиця 6.2 – Вибір вимірності в якій буде проектуватись рух робота

Мета тесту:	**Перевірка функції «Вибір вимірності структури»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснути кнопку «2D» чи «3D»
Очікуваний результат:	З’явиться надпис «2D» чи «3D» над кнопками відповідно до обраного варіанту
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.3 – Перевірка заповнюваності вартості переходу між вершинами структури

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести розмірність структури та не вказати вартості переходів між вершинами структури та натиснути на кнопку «Sumbit»

Продовження таблиці 6.3

Очікуваний результат:	З’явиться попереджувальний надпис «Fields must contain only numbers»
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.4 – Перевірка заповнюваності розмірності структури

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести вартості переходів між вершинами структури та не вказати розмірність структури та натиснути на кнопку «Sumbit»
Очікуваний результат:	З’явиться попереджувальний надпис «Fields must contain only numbers»
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.5 – Перевірка отримання вхідних даних системою

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**

Продовження таблиці 6.5

Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	Розмірність структури та товщини стінок
Схема проведення тесту:	Ввести вартості переходів між вершинами структури та вказати розмірність структури та натиснути на кнопку «Sumbit»
Очікуваний результат:	З’явиться попереджувальний надпис «SUMBMITTED INPUT»
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.6 – Перевірка візуально зображення в результаті генерації структури

Мета тесту:	**Перевірка функції «Генерація та зображення структури»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснути на кнопку «Generate field»
Очікуваний результат:	З’явиться візуальне зображення північно-західної частини структури
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.7 – Перевірка отримання координат пошуку системою

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	Координати кімнати А та координати кімнати В
Схема проведення тесту:	Отримання від користувача початкової та кінцевої координат вершин структури та натиснення кнопки «Submit»
Очікуваний результат:	З’явиться повідомлення «SUBMITTED COORDINATES»
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.8 – Перевірка введення неправильних координат пошуку системою

Мета тесту:	**Перевірка функції «Перевірка заповнюваності»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	Координати кімнати А та координати кімнати В

Продовження таблиці 6.8

Схема проведення тесту:	Отримання від користувача початкової та кінцевої координат вершин структури невідповідного типу та натиснення кнопки «Submit»
Очікуваний результат:	З’явиться повідомлення «FIELDS MUST CONTAIN ONLY NUMBERS»
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.9 – Перевірка коректності пошуку кількості взагалі можливих шляхів

Мета тесту:	**Перевірка функції «Пошук кількості шляхів»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення кнопки «Number of ways»
Очікуваний результат:	З’явиться в кожній координаті на візуальному зображені зеленим кольором кількість взагалі можливих шляхів
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.10 – Перевірка коректності пошуку оптимального шляху

Мета тесту:	**Перевірка функції «Пошук оптимального шляху»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення кнопки «Dijkstra» або «Ant»
Очікуваний результат:	З’являться на візуальному зображені різнокольорові стрілки між вершинами структури, що вказуватимуть оптимальний шлях
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.11 – Перевірка коректності роботи навігації

Мета тесту:	**Перевірка функції «Навігація»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення будь-якої стрілки із наведених для навігації стрілок
Очікуваний результат:	Візуальне зображення структури змінить фокус на одну із координат в залежності від обраної користувачем стрілки

Продовження таблиці 6.11

Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.12 – Перевірка коректності роботи перемикачів графічного відображення елементів структури

Мета тесту:	**Перевірка функції «Графічні перемикачі»**
Початковий стан КЗ	Відкрита головна «сцена»
Вхідні данні:	
Схема проведення тесту:	Натиснення будь-якого з перемикачів відображення графічних об’єктів
Очікуваний результат:	При ввімкнені відповідні графічні об’єкти з’являться, при вимкнені графічні об’єкти зникнуть
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Таблиця 6.13 – Перевірка коректності роботи системи в випадку коли користувач навігацією перетне крайні межі структури

Мета тесту:	**Перевірка функції «Навінація»**
Початковий стан КЗ	Відкрита головна «сцена»

Продовження таблиці 6.13

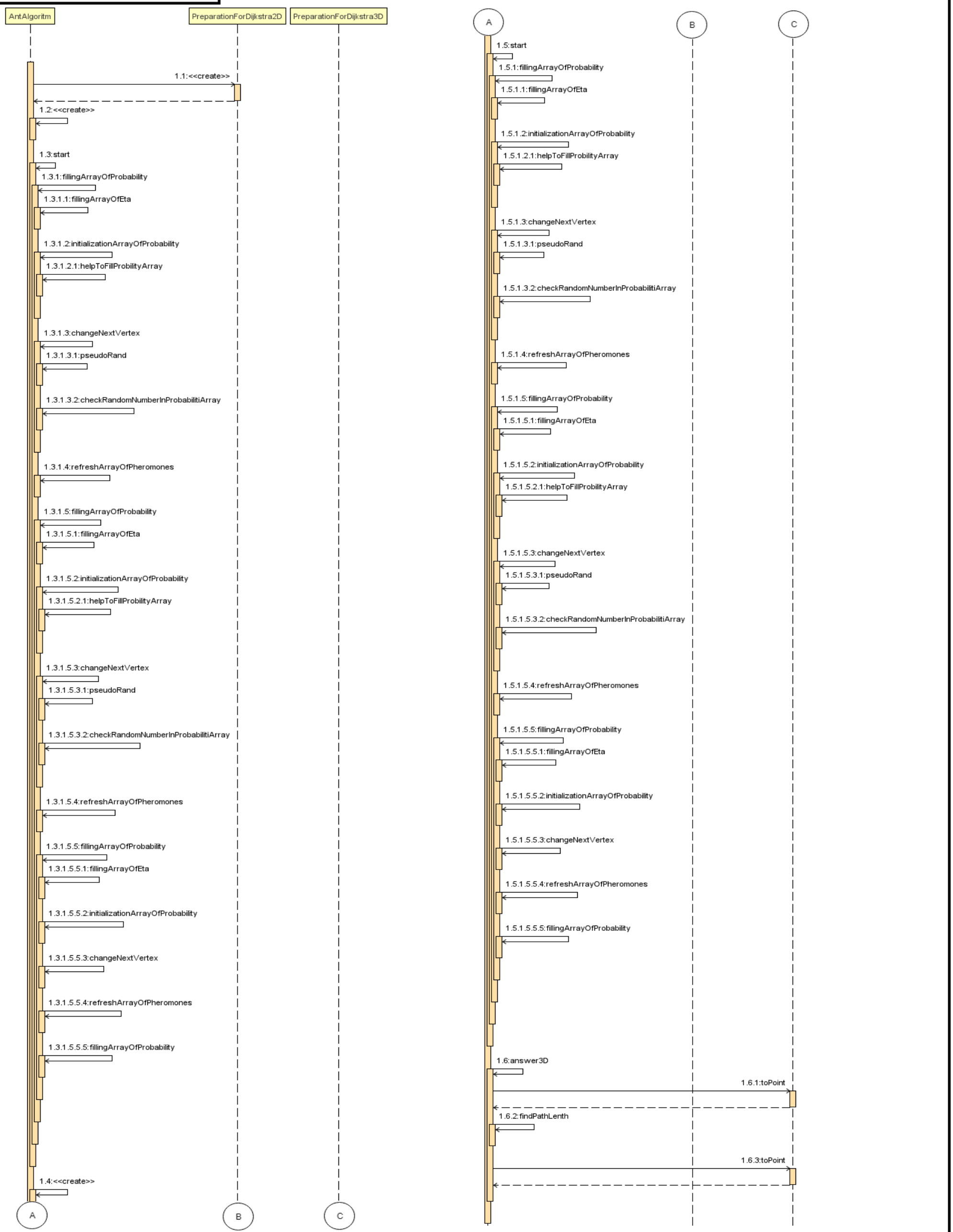
Вхідні данні:	
Схема проведення тесту:	Натискання користувачем однієї і тієї ж кнопки навігації до тих пір поки візуальне зображення структури не перетне межі структури
Очікуваний результат:	Відображення пустого зображення структури
Стан КЗ після проведення випробувань:	Відкрита основна «сцена»

Графічний матеріал
до дипломного проекту

на тему: Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору

Київ – 2019 року

ДП ІС-5221.1181-с.ССП



Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Поліщук А.О.		
Перев.		Солдатова М.О.		
Т. Кон.				
Н. Кон.		Халус О.А.		
Затв.		Солдатова М.О.		

ДП ІС-5221.1181-с.ССП

Схема структурна послідовності

Лит.	Маса	Масштаб

Аркуш 1 | Аркушів 1

Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору

КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-52

ДП ІС-5221.1181-с.ССК

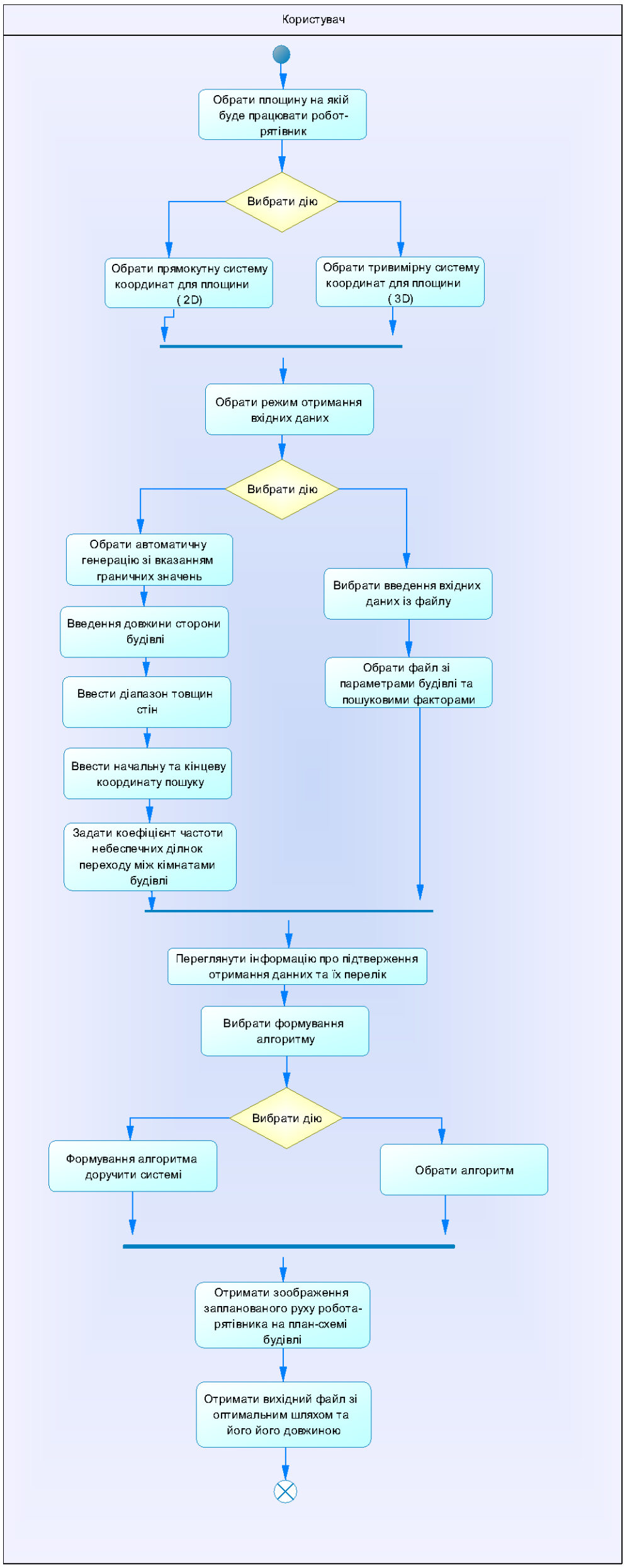
					ДП ІС-5221.1181-с.ССК
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення
Розробив		Поліщук А.О.			
Перевірив		Солдатова М.О.			
Т. кон.					Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору
Н. кон.		Халус О.А.			
Затвердив		Солдатова М.О.			

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-52

ДП ІС-5221.1181-с.ССВ

					ДП ІС-5221.1181-с.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання	Літера / Маса / Масштаб
Розробив		Поліщук А.О.				
Перевірив		Солдатова М.О.				Аркуш 1 / Аркушів 1
Т. кон.					Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної стрруктури замкнутого робочого простору	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Солдатова М.О.				



Зм.	Арк.	№ докум.	Підп.	Дата	ДП ІС-5221.1181-с.ССД		
Розроб.		Поліщук А.О.			Схема структурна діяльності	Лит.	Маса / Масштаб
Перев.		Солдатова М.О.				Аркуш 1	Аркушів 1
Т. Кон.					Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. Кон.		Халус О.А.					
Затв.		Солдатова М.О.					

ДП ІС-5221.1181-с.ССК

					ДП ІС-5221.1181-с.ССК
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення
Розробив		Поліщук А.О.			
Перевірив		Солдатова М.О.			
Т. кон.					Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору
Н. кон.		Халус О.А.			
Затвердив		Солдатова М.О.			

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-52

ДП ІС-5221.1181-с.ССР

					ДП ІС-5221.1181-с.ССР	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна розгортання обчислювальних вузлів	Літера / Маса / Масштаб
Розробив		Поліщук А.О.				
Перевірив		Солдатова М.О.				Аркуш 1 / Аркушів 1
Т. кон.					Автоматизація пошуку оптимального шляху робота рятівника в умовах невизначеної структури замкнутого робочого простору	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Солдатова М.О.				